Action Script 3.0

العنوان : 11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية

تليفون وفاكس : 4838326 (03)(2+)

للاستعلام والمبيعات : 01001634294 (2+)

URL: www.daralbraa.com
Email: info@daralbraa.com

أيمن محبوب

ayman_mahboub@yahoo.com

الناشر:	دار البراء لنشر وتوزيع الكتب العلمية
رئيس مجلس الإدارة:	إبراهيم محمد إبراهيم زبير
اسم الكتاب:	Action Script 3.0
المؤلف:	**أيمن رمضان علي محبوب**
رقم الإيداع:	1927/2013
الترقيم الدولي:	**978-977-723-007-0**
المقاس:	24 x 17
عدد الصفحات:	136
العنوان:	11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية
تليفون وفاكس:	(+2)(03) 4838326
للاستعلام والمبيعات:	(+2) 01001634294
البريد الالكتروني:	info@daralbraa.com
الموقع:	**WWW.DarAlbraa.com**

إهداء

إهداء إلى عبد الرحمن وجنى قرتا عينى ..

إهداء إلى أصدقائي و زملائي الغاليين ..

إهداء إلى طلاب العلم في كل مكان ..

إهداء إلى محبي الفلاش وراغبي تعلم البرمجة ..

أهديهم وأهديكم جميعا ذلك العمل الذي شرفت به ..

وأسأل الله أن يكون فية الفائدة والنفع للجميع ..

مقدمة

Action Script :

الأكشن هي لغة برمجية خاصة ببرنامج الفلاش ، عن طريقها نكتب أمرا أو عدة أوامر تجعل فيلم الفلاش يقوم بعمل أشياء معينة (Actions) أثناء التشغيل . ويتم كتابة هذه الأوامر على شكل جمل لها صياغة خاصة في لوحة Actions ، وعند تصدير العمل بصيغة فيلم (SWF) فإن الفيلم يقوم بالعمل بناء على تلك الأوامر .

وهذه اللغة مبنية على مجموعة من القواعد تسمى الإيكيما إسكربت وهي مشتق من **E**uropean **C**omputer **M**anufacturers **A**ssociation وهى رابطة مصنعي الكمبيوتر الأوروبية وهى رابطة متخصصة في وضع مقاييس مواصفات عدة لغات برمجية .

تاريخ Action Script :



(جوناثان جاي)

- **في يناير 1993** لم يكن استخدام الحاسب والإنترنت منتشراً كما هو الآن ، بدأ جوناثان جاي ، تشارلي جاكسون ، وميشيل ويلش بتكوين شركة برمجيات صغيرة تسمى **FutureWave** للبرمجيات وإنشاء أول منتج ، **"SmartSketch"** ، وقد صمم **"SmartSketch"** - وهو برنامج للرسم على الكمبيوتر الذي يعمل بالقلم على نظام التشغيل **PenPoint -** لجعل رسومات الحاسوب بسيطة مثل الرسم على الورق إلا أن هذا البرنامج فشل فشلا ذريعاً ولم يلاقى النجاح المرجو منه .
- عندما بدأت شبكة الإنترنت في الازدهار بدأت **FutureWave** بالتفكير في تحقيق إمكانيات عمل الرسوم المتحركة على شبكة الإنترنت وهذا يعد طفرة كبيرة في وقتها حيث كانت شبكة الانترنت تعتمد فقط على عرض لنصوص وصور بسيطة ليس أكثر ، وفي عام 1995 قامت شركة **FutureWave** بتعديل برنامجها **SmartSketch** بإضافة تقنيه الإطار تلو الإطار للرسوم المتحركة وإعادة إصداره تحت اسم **FutureSplash** للرسوم المتحركة .

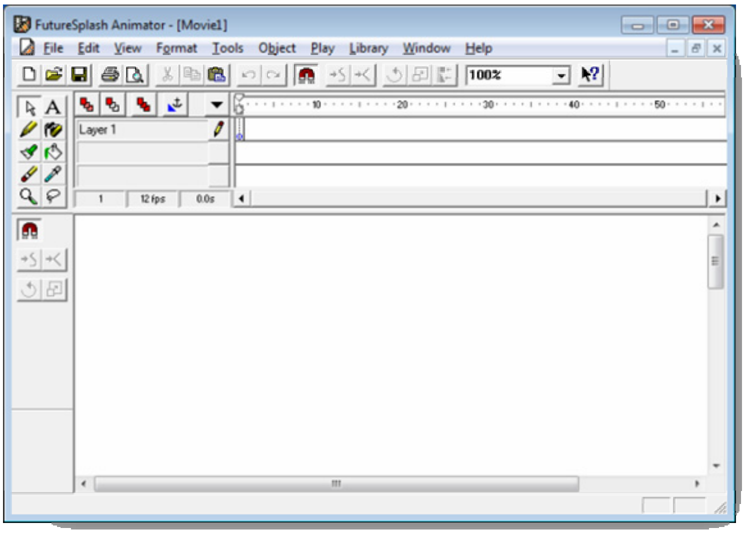


(FutureSplash)

بحلول ذلك الوقت كانت الشركة قد قامت بتعيين المبرمج روبرت تاتسومي والفنان آدم جروفكيسيك و المتخصص في العلاقات العامة رالف ميتمان ، تاتسومي وجاي عملوا معا لتطوير المنتج وعرض المنتج على شركة أدوبي ومن الطريف أن شركة أدوبى رفضت هذا العرض ورأت أن البرنامج لا يرقى لبرامج الجرافيك الذي يجعلها تقدم على شراءه .

- **فـي ابريـل عـام 1996** ، اشـترت شـركة " ماكروميـديا " البرنـامج وأطلق في السوق في ديسمبر 1996 تحت اسم **Flash** المستوحى من كلمتي **"Splash"** و **"Future"** و لـم تغير فيـه أي مواصـفات ، مجـرد تغيير أسم فقط ، حيث كان يحتوى البرنـامج علـى أدوات التحريـر الأساسية ، و إطار زمني ، كما في الصورة

- التالية :



(شكل لواجهة FutureSplash البسيطة)

- **في يونيه 1997** قامت ماكروميديا بتطوير البرنامج وقامت بإصدار النسخة الثانية من **Flash 2** ومعه مشغل الفلاش **Flashplayer 2** ، وفي هذا الإصدار كان أول ظهور للمكتبة داخل البرنامج بالإضافة لبعض التعديلات على الإطار الزمني والأدوات .

- **في 31 مايو 1998** قامت ماكروميديا بإصدار **Flash 3** و معه مشغل الفلاش **Flashplayer 3** وكان من أهم مميزاته :
  - ظهور مفهوم ال **movieclip .**

- ظهور مفهوم **stand alone player** بمعنى أمكانية إخراج ملف الفلاش في ملف منفصل يعمل دون الحاجة لمشغل فلاش وهو الملف ذو الامتداد **exe** .
- بداية ظهور الأكشن ليس كلغة كما نستخدمها الآن ولكنها كانت مجرد أوامر بسيطة على شريط الزمن مثل **play** , **gotoandStop** , stop , **gotoandStop** .



Garry Grossman

وكان لظهور المبرمج العبقري جاري جروثمان في هذا القترة الفضل في ظهور لغة الأكشن أسكريبت وكانت قريبة في شكلها من الجافا أسكريبت .

- **وفي 15 يونيه 1999** تم إصدار **flash 4** و معه **Flashplayer 4** ، وكان بداية لاستخدام الاكشن سكريبت و السماح بدخول ملفات **streaming MP3** .

- **في 24 أغسطس** 2000 تم إصدار **flash 5** و معه **Flashplayer 5** ، وكان نقلة كبيرة في عالم الفلاش حيث بداية لاستخدام الآكشن كلغة برمجة وظهور **Action script 1** وبداية تدعيم ملفات **xml**

- **في 15 مارس 2002** تم إصدار **flash mx** ومعه **Flashplayer 6**

مميزاتة :

- تدعيم ملفات الفيديو **flv** .
- ظهور مفهوم ال **components**
- أمكانية الرسم من خلال الأكشن سكرييت

- **في 10 سبتمبر 2003** تم إصدار **flash mx2004** وفي هذا الوقت بدأت شركة ماكروميديا أصدار نسختين أحدهما نسخة عادية والأخرى **professional**

مميزاتة :

- ظهور **Action script 2** وهو منظم أكثر لتدعيمة **Object Oriented Programming** .
- تدعيم **web services** .
- تطوير كبير في **components** واصدار نسخة أحدث عن سابقتها تحتوى على **Media Playback components** والذى يمكننا من تشغيل ملفات والتحكم في ملفات ال **MP3**

- **وفي 13 سبتمبر 2005** تم إصدار **Flash Professional 8** وكان ينظر لهذا الإصدار على أنه تحديث للمصممين المستخدمين للفلاش كما كان ينظر للإصدار **flash mx2004** على أنه تحديث للمبرمجين المستخدمين للفلاش .

مميزاته :

- ظهور خاصية ال **Filters** وإمكانية اسستخدامها من خلال الكود
- ظهور ال **Flash lite** لعمل تطبيقات الفلاش على الموبايل

في هذا الوقت تطور برنامج الفلاش تطور لفت أنظار الشركات الكبيرة مثل شركة **Adobe** والتى سبق ورفضت البرنامج في بدايات ظهوره إلا أنها أدركت أهميته ضمن مجموعة الجرافيك التي تحويها لذا قامت شراء شركة **Macromedia** بسعر 3.6000.000.000 بليون دولار

- **وفى ابريل 2007** ظهر أول نسخه للفلاش من شركة **Adobe** من **Adobe Flash Cs3**

مميزاتة :

- ظهور **Action script 3** .
- تطوير مساعد أكشن سكريبت **Script Assist** لإستعماله مع **Action script 3** .

- تكامل البرنامج مع برامج **Adobe** الأخرى .
- تطور كبير في أدوات الرسم .
- امكانية تشغيل أنواع اخرى من ملفات الفيديو .
- المزيد من الإضافات في مجال عمل **Software** للموبايل
- للتحكم بشكل أفضل في تغيير حجم الموفى كليب
- أضيف المزيد من الجودة إلى الملفات التي يتم تصديرها بصيغة **QuickTime** ذو المتداد **mov** .

- **في أكتوبر 2008** ظهر **Adobe Flash CS4** ومعه **Flashplayer 10**

مميزاته :

- إضافة تقنية **Inverse Kinematics** وهى تقنية الحركة من خلال ال **Bone tool** أي العظم .
- التعامل مع الأشكال وحركتها بطريقة تحاكى ال **3D** .
- دعمه للكتابة من اليمين للشمال في ال **Flashplayer** .
- تعامل البرنامج مع نوع جديد من الفيديو ذو الأمتداد **F4v** .
- التحكم باحترافية بأي حركة تقوم بها وذلك عن طريق لوحة **Motion Editor**.

- **في ابريل 2010** تم إصدار **Adobe Flash CS5** وكان من أهم مميزات هذا الأصدار :

- دعمه لتطبيقات **iPhone** .

- **Code Snippets** وهى لوحة خاصة بمجموعة من الكواد الجاهزه للأستخدام .

- إضافة نوع جديد من الكتابه وهو **TLF** يمكننى من الكتابة من اليمين للشمال و بهذا تكون مشكلة الكتابة باللغة العربية قد حلت.

- تحسين الحركة على **Inverse Kinematics** من خلال الخاصية **spring** .

- إمكانية **embedded fonts** لدمج نوع النص داخل ملف الفلاش لضمان عدم تغييره من جهاز لأخر .

- **وفى 2011** تم اصدار **Adobe Flash CS5.5** كتحديث للنسخه السابقة من حيث دعم الفلاش لتطبيقات **iPhone** .

- **وفي ابريل 2012** تم اصدار النسخة **Adobe Flash CS6** والتى من أهم مميزاتها :

- **Generation Sprite Sheet** والتى تمكني من عمل تصدير لل **Symbols** ومتابعة تسلسل حركتة وهذا ساعد بشكل كبير في تحسين تجربة الألعاب .

- دعم تطبيقات **Android** و **iOS** بشكل أوسع

- **Adobe AIR mobile simulation** محاكاة التفاعلات المشتركه لتطبيقات الهاتف المحمول مثل اتجاه الشاشة، والإيماءات التي تعمل باللمس.

ما سبق كان مقدمة عن الفلاش وتاريخة حتى ندرك اثناء دراستنا للاكشن أن ما ندرسه حاليا كان نتاج لتعب وجهد الكثير من المطوريين للبرنامج على مدار السنين ، ليكون بين أيدينا الآن برنامجا سهل في دراسته وفهمه وتطبيقاتة .

مع قليل من الوقت وبسيط من الفهم وكثير من التدريب والعمل تصل إلى مبتغاك بأذن الله .

أيهما أفضل Action Script 2 أم Action Script 3 ؟

أن الاختلاف بين **Action Script 2** و **Action Script 3** هو الاختلاف بين جيلين من البرمجة ونظرا للتطور الهائل في برنامج الفلاش وتحديث أدواته بشكل مذهل كان لا بد من ظهور **Action Script 3** . وتستطيع أنت أن تجيب على السؤال السابق من خلال استفساري التالي :

- إذا كان ما تحتاجه من الفلاش هو حركات بسيطة وبعض الأكواد التي تتحكم في الحركة مثل **gotoandStop , gotoandStop** فعليك أن تختار تعلم وفهم **Action Script 2** وأنصحك أن لا تكمل قراءة كتابى هذا ، كما لا تنتظر أي تطور في عملك حيث أنك لن تستطيع مواكبة التطور التقنى والذي أصبح يعتمد على الفلاش و **Action Script 3** في مجالات عدة.
- أما إذا كنت من الطموحين والساعيين أبداً إلى التطور والنجاح فعليك إذا أن تختار تعلم **Action Script 3** ، ولا تخف أبداً من فكرة البرمجة فالأمر ليس كما تتخيل من صعوبة و لا نحتاج منك سوى الفهم الجيد

والاستيعاب لفصول الكتاب و الكثير من التدريب والعمل ليرقى عملك إلى صفوف المحترفين الناجحين في عملهم وأشكرك على اختيارك لكتابي هذا فقد راعيت فيه السهولة واليسر في صياغة فصوله والشرح المبسط حتى يصبح لك الأمر ممتعا في الدراسة .

ولي تخيل معين في الفرق بين **Action Script 2** و **Action Script 3** وهو انك وأنت تتعامل مع برنامج الفلاش تكون كمدرس الفصل الذي يريد أن يعطى تعليماته للطلبة ففي وضعية **Action Script 2** أنت مضطر أنت تذهب إلى كل طالب وتملي عليه بعض الأوامر ، فتخبر هذا: أنت عليك فعل ذلك وتخبر أخر :أنت لا تفعل هذا ، وهكذا إذا أردت أن ينفذ جميع الطلاب أوامرك عليك أن تذهب لكل طالب بمفرده وتخبره بما يجب عليه فعله وإذا أردت تعديل تعليماتك فعليك أن تعيد الكره مرة أخرى وتذهب إلى مكان كل طالب لتخبره أنك قد غيرت قرارك بالنسبة له ، يبدو الأمر ممل وصعب وغير عملي بالمرة .
تعالى لنتخيل كونك مدرس في وضعية **Action Script 3** كل ما عليك هو كتابه الأوامر أجمالا في ورقة واحدة وتعليقها في أول الفصل وإذا كنت تريد أعطاء أمر معين لطالب معين يمكنك كتابة أسم الطالب في الورقة وأعلامه بالأوامر المطلوب منه تنفيذها ، وأما إذا كان هناك تعديل فما عليك سوى القيام بذلك في ورقة الأوامر المعلقة في الفصل ، ألا يبدو الأمر في الوضعية الثانية أسهل بكثير و أفضل وأكثر حكمة ، والأروع من ذلك أنك يمكنك استخدام نفس الورقة بأوامرها لفصل أخر بل لفصول أخرى دون جهد جهيد .

الفصل الأول
Action Script Panel

- أن أول شئ من البديهي أن نفعله أن نقوم بفتح ملف جديد من النوع Action Script 3 وإذا ما فتحنا الملف من النوع Action Script 2 فأننا يمكننا تعديل ذلك من لوحة Properties والخاصة بخصائص الملف المفتوح .

مكان كتابة الكود :

هناك طريقتين لكتابة الكود في Action Script 3 وهما :

أولا - كتابة الكود في ملف Action Script منفصل يتم اختياره عند فتح ملف

جديد و يجب بالطبع إدراج ملف الأكشن مع ملف الفلاش(SWF) ليعمل

كما هو مطلوب .

ثانياً - كتابة الكود في أي key frame في ال Time Line وذلك من خلال

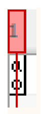
نافذة الـ Actions والتى تأتى بإحدى طريقتين :

1 - عن طريق الذهاب إلى قائمة Window ثم اختيار Actions .

2 - عن طريق الضغط على مفتاح F9 **من لوحة المفاتيح .**

ملاحظات:

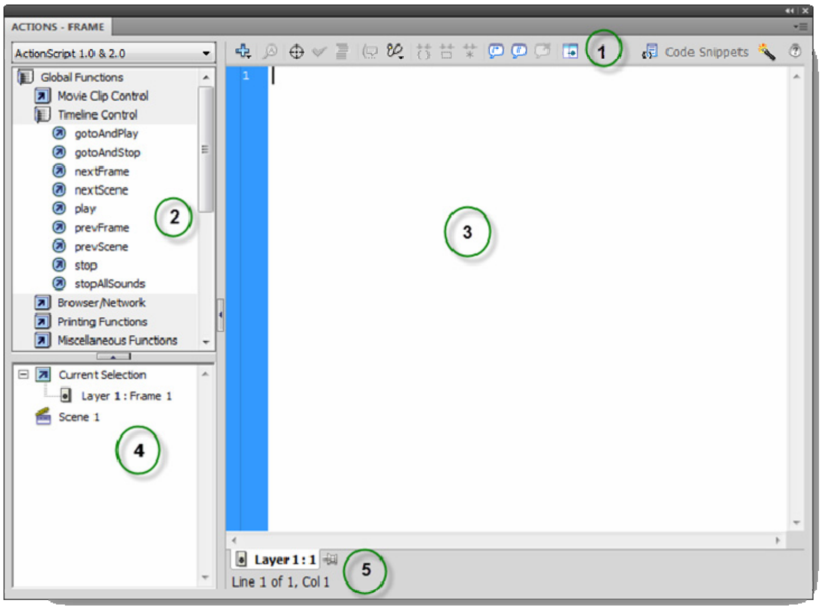
1- عندما يتم كتابة كود أكشن سكريبت فإن البرنامج يضع حرف a على الإطار الذي تم وضع الأكشن عنده فيكون بهذا الشكل .



2- وكشكل تنظيمي يفضل وضع الطبقة الخاصة بالأكشن في أعلى الطبقات .

التعرف على لوح Actions :

والآن دعونا نلقى نظرة على لوح Actions ، حيث يمكننا تقسيمه إلى الأقسام الخمس التالية :



القسم الأول - شريط أدوات لوح الأكشن:



1- تماما مثل صندوق الاكواد يمكنك من خلالها البحث عن كود معين واضافته .

2- أيقونة البحث عن كلمة في الكود الذي كتبته واستبدالها.

3- أيقونة مسار الكائن فقد يحدث أن يكون لديك كائن داخل آخر دخل آخر فهذه الأيقونة تزودك بشجرة توضح علاقة الكائنات ببعضها فيمكنك اختيار كائن معين وهذه الأيقونة تكون لك مساره .

4- أيقونة التدقيق النحوي والتي تتأكد من سلامة الاكواد المكتوبة .

5- أيقونة التنسيق الذاتي حيث يقوم بمحاذاة الاكواد وتنسيقها حيث أن لغة أكشن سكريبت مثلها مثل أي لغة لها بعض القواعد في طريقة كتابتها وبالضغط على هذه الأيقونة يتم تصحيح التنسيق وفقا للقواعد .

مثال : الكود التالي تمت كتابته هكذا

```
import flash.net.URLRequest;

 link_btn.addEventListener(MouseEvent.CLICK, go);

 function go(event:Event):void
{
var linking:URLRequest = new URLRequest("http://www.yahoo.com");
               navigateToURL(linking);
}
```

ثم بعد الضغط على أيقونة التنسيق التلقائي أصبح الكود هكذا

```
import flash.net.URLRequest;

link_btn.addEventListener(MouseEvent.CLICK, go);

function go(event:Event):void
{
    var linking:URLRequest = new URLRequest("http://www.yahoo.com");
    navigateToURL(linking);
}
```

6- أيقونـة التلميحـات أثنـاء كتابـة الكـود يقـوم المحـرر بإظهـار بعـض التلميحات حول طريقة كتابة واستعمال دالة أو خاصية معينة .

7- أيقونة خيارات التصحيح مثلا إزالة أو إضافة نقاط توقف عند سطر معين لاختبار البرنامج .

8- أيقونـة الاختـزال ، الأولـى للاختـزال سـطر واحـد بمعنـى تجميعـه وإختزاله إلى أيقونة صغيرة معلمة بعلامة + لتوفير مساحة ، الثانية اختزال عدة اسطر ، الثالثة للتراجع وعرض كل ما تم إختزاله

9- أيقونة التعليقات ،الأولى لإضافة تعليق عبارة عن عدة اسطر ،الثانية لإضافة تعليق عبارة عن سطر واحد ،والثالثة لإزالة التعليق بعد تنشيطه .

10- أيقونة إظهار أو إخفاء صندوق الأكواد .

11- Code Snippets :

وهى لوحة خاصة بمجموعة من الكواد الجاهزة للاستخدام ، والتى تسهل الكثير من العمل وتوفر عناء كتابة كود كبير مكون من عدة أسطر وتجنبك الخطأ في كتابة الكود ، فعلى سبيل المثال إذا كنت تريد عمل زر عند النقر عليه يذهب إلى رقم Frame معين ، هنا كل ما عليك هو عمل الزر في المكان المطلوب ثم قم بوضع أسم له أمام خانة Instance name من لوحة Properties وحتى لو لم تسمية فبمجرد الوقوف على الزر والنقر على Code Snippets و اختيار من مجموعة Timeline Navigation الكود المطلوب وهو Click to Go to Frame and Play هنا

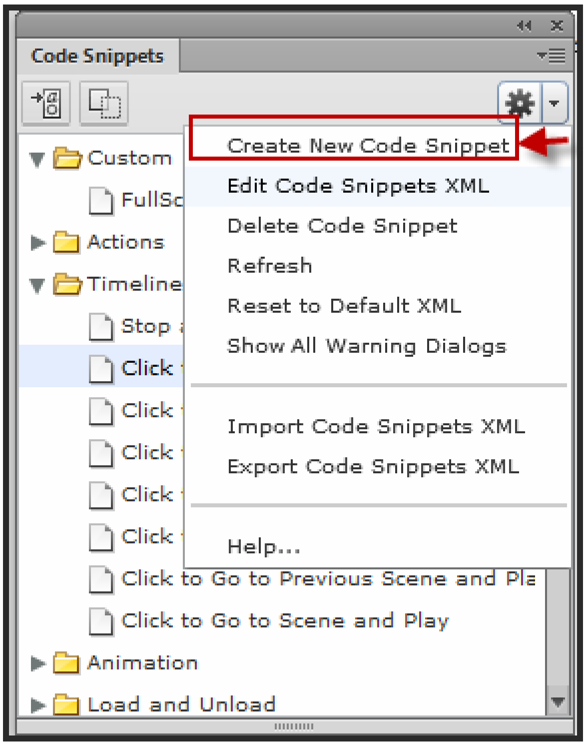
سوف يقوم الكود بإعطاء أسماً افتراضيا وليكن button_1 ويعطيك الكود المطلوب كالتالي :

```
button_1.addEventListener(MouseEvent.CLICK, fl_ClickToGoToAndStopAtFrame);

function fl_ClickToGoToAndStopAtFrame(event:MouseEvent):void
{
	gotoAndStop(5);
}
```

ويمكنك تعديل رقم الفريم الذي تريد الذهاب له وتعديل أسم الزر بكل سهولة .

ويمكنك توفيرا للوقت من عمل مكتبة من الأكواد والاحتفاظ بها داخل فولدر خاص بك وضمن مجموعات Code Snippets وذلك من الخيار Code Snippets Create New Code Snippet



12- مساعد أكشن سكريبت وهى تحول محرر كتابة الكود إلى وضعية سطر بسطر أي لا يمكنك كتابة أو تعديل أو إضافة أي كود في منطقة كتابة الأكواد ، فقط تحدد السطر ثم تضيف دالة أو خاصية من صندوق الأكواد .

وعند النقر على تلك الأيقونة تبرز منطقة فوق منطقة كتابة الكود ، يوجد بتلك المنطقة كل الخيارات المتعلقة بالأمر الذي اخترته ، ببساطة قم بتحديد أي سطر من أسطر الكود Script لتجد المساعدة الخاصة به، وعندئذ بدلا من تعديل الكود بالكتابة مباشرة ، يمكنك عمل التعديل في الخيارات المتاحة وعندها يقوم البرنامج بالنيابة عنك بتعديل الكود وفقا لذلك .

القسم الثاني - فئات Actions المختلفة :

وهو يعتبر قاموس يحوى جميع الأكواد الخاصة بالكشن سكريبت ويمكنك الوصول إليها أيضا بالنقر على الأيقونة  في شريط أدوات الأكشن .

القسم الثالث - منطقة الـ Script

أو كتابة الكود الخاص بلغة Action Script .

القسم الرابع - منطقة التنقل

وهى تحتوى على أماكن وجود الـ Actions التي تستعملها في فيلمك الحالي بما فيها Current Selection ، أي المكان الحالي حيث يمكنك التنقل بسهولة بين أكواد الأكشن ، فقد يوجد كود الأكشن بأحد الإطارات ، وكلما قمت بإسناد أكشن إلى إطار معين ، يتم على الفور إضافة ذلك الإطار إلى منطقة التنقل ، وبمجرد النقر على أي Frame داخل منطقة التنقل ، يتم عرض كود الأكشن المرتبط بذلك ال Frame داخل منطقة الـ Script .

القسم الخامس - شريط المعلومات

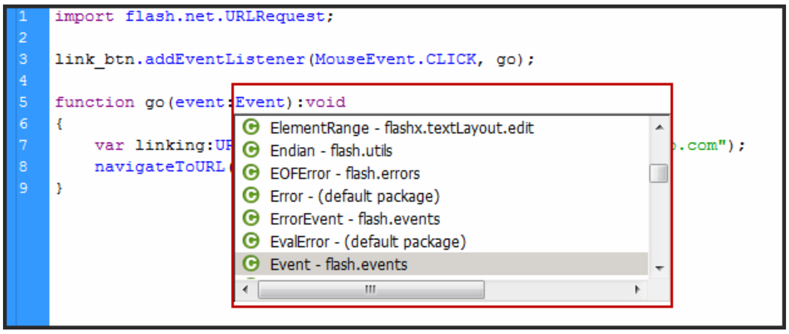
يعرض معلومات عن رقم السطر الحالي وعدد الأسطر الإجمالية وما هو الرمز المتعلق بهذا الكود ، كما يمكن أن يضم الشريط أيضا نوافذ مصغرة لأكواد الأكشن داخل العمل .

تلميحات أكشن سكريبت :

بالإضافة إلى المساعدة الممتازة التي يقدمها مساعد الأكشن  ، يمكننا أيضا الاستعانة بالتلميحات التي يقدمها البرنامج أثناء كتابة الكود ، حيث تظهر

تلميحات تحتوى أحيانا على الصيغة العامة للأكشن الذي تقوم بكتابته - كى تقوم بكتابته بالشكل الصحيح - وأحيانا يكون التلميح في صورة قائمة منسدلة يمكنك اختيار أحد عناصرها ، حيث تحتوى تلك القائمة على أسماء الخصائص والوظائف الممكنة

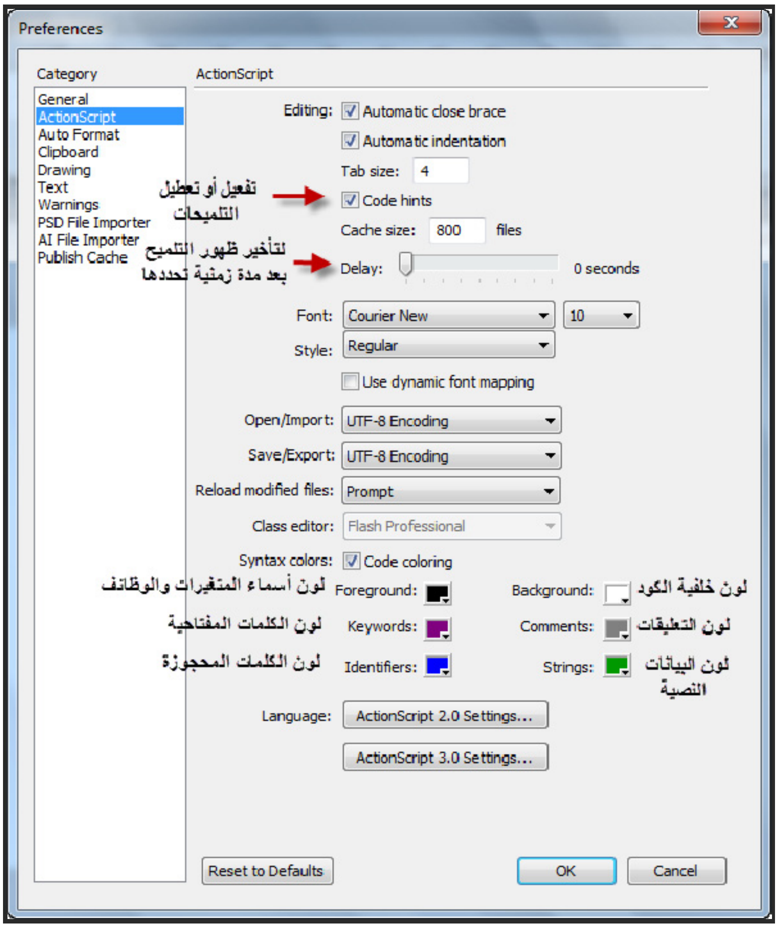
في المثال التالي مثال على التلميح في شكل قائمة منسدلة تظهر للمساعدة في اختيار أحد الخصائص اللازمة حدوث الحدث .



تعديل الخيارات الخاصة بلوحة الأكشن :

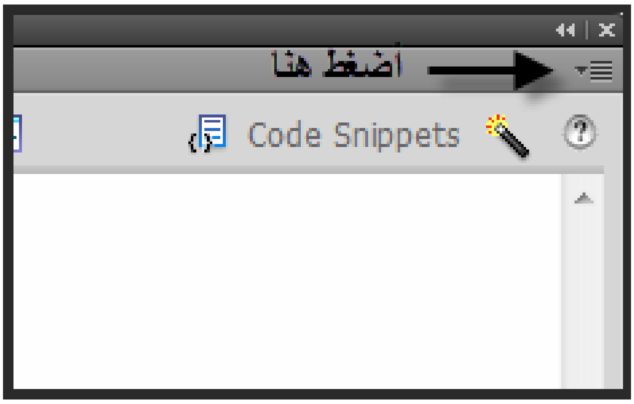
من القائمة Edit ← Preferences أو من لوحة المفاتيح (Ctrl + u) ثم اختيار التبويب Action Script .

حيث يمكننا تعديل بعض الخيارات في نافذة أكشن سكريبت كما هو موضح بالصورة



استيراد وتصدير الأكشن :

انقر على هذه الأيقونة في الركن الأيمن العلوي من لوح أكشن



حيث تظهر قائمة بها خيارات عديدة ، والذي يهمنا الآن هي الخيارات الخاصة باستيراد وتصدير الأكشن



حيث يمكن اختيار Export Script لتصدير الكود الحالي , ويكون الملف الناتج بصيغة as ، وهو بالطبع اختصار ل Action Script ، ويكون بهذا الشكل



ومن نفس القائمة المذكورة يمكننا اختيار Import Script لاستيراد ملف أكشن سكريبت .

إضافة ملاحظات:

يمكنك إضافة ملاحظات في نص السكريبت تكون بمثابة مرجع لك إذا أردت عمل بعض التعديلات فيما بعد أو لتتذكر المقصود من الأمر البرمجي الذي كتبته ، فبالتأكيد أنت تعرف ذلك الآن ، ولكنك بعد شهور أو سنوات ستنسى بالطبع ، كما أن ذلك مفيد جدا إذا كنت ضمن مجموعة تقوم بعمل كبير وكل مصمم في المجموعة يقوم بجزء معين في الفيلم ، إذن ملاحظاتك ستمكن زميلك من فهم المقصود من الأوامر المكتوبة ، ولكتابة أي ملاحظة يجب أن تبدأ بشرطتين مائلتين هكذا **//** عندها يعرف البرنامج أن ما يأتي بعد هذه العلامة لا يدخل في نطاق الأوامر البرمجية . وهذا الأسلوب شائع في كل لغات البرمجة أو البرامج التي لها لغات برمجة (مثلا في فيجوال بيسك تبدأ الملاحظة بالعلامة ')

ويمكنك كتابة الملاحظات بالعربية أو الإنجليزية .

```
//عند النقر على الزر يتم فتح صفحة ياهو
```

وإذا كان التعليق يتكون من عدة أسطر ، فيمكننا استخدام الشرطة المائلة متبوعة بالنجمة في بداية التعليق ، ثم النجمة متبوعة بالشرطة المائلة في نهاية التعليق

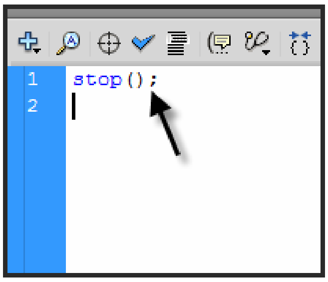
```
/* Increase velocity of
the movie clip at that point
then decrease by five every minute*/
```

استخدام الاختصارات في كتابة الكود:

يمكننا استخدام الاختصارات التي يقدمها البرنامج ، فمثلا لإدخال الأكشن trace داخل الكود يمكننا استخدام الاختصار Esc + tr
حيث نستمر في الضغط على مفتاح Esc في لوحة المفاتيح أثناء كتابة حرف t ثم حرف r ، فيتم إضافة كلمة trace داخل الكود

نهاية الجملة في أكشن سكريبت :

تنتهي الجملة في أغلب الأحوال بفاصلة منقوطة (Semicolon) كما الصورة التالية، ويجب الانتباه إلى أننا نتحدث عن الجملة وليس السطر حيث أن الجملة قد تتكون من عدة أسطر أن كانت طويلة .



ما سبق كان استعراض للوحة الأكشن وبعض المفاهيم المتعلقة بكتابة الكود وما علينا الآن سوى الدخول إلى عالم ال Action Script 3 .

الشق التصميمي في برنامج الفلاش:

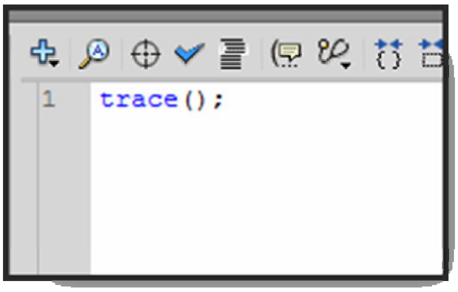
أن برنامج الفلاش يعتمد في الأساس على الرسم وتناسق الألوان و التناغم في الحركة بين مكونات التصميم مستخدمين أدوات البرنامج التي تطورت بشكل

هائل على مدار تحديث البرنامج ، وقد قمنا بشرح البرنامج وأدواته في كتب أخرى يمكنك الاطلاع عليها ودراسة الأدوات وكيفية الرسم بسهولة .
ولسوف نتعرض في كتابنا هذا للشق البرمجي فقط مكملين بذلك عملنا في التصميم الجيد الاحترافي .

: Trace

- يهمنا أن نتعرض في بداية شرحنا لوظيفة برمجية هامة (Function) تسمى trace مهمتها عرض قيمة رقمية أو نصية داخل لوحه output حيث تستخدم لاختبار مدى صحة الكود .

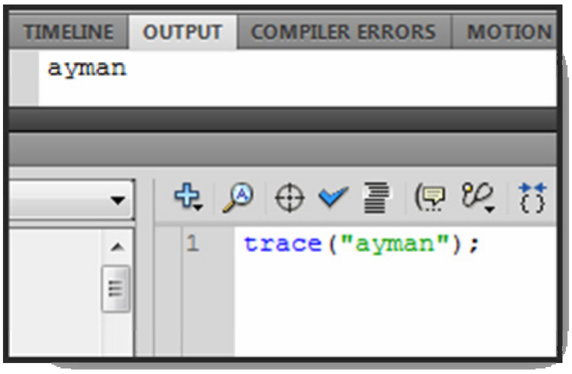
لإستدعاء Function يجب أن أناديها باسمها ، ولاستدعاء الوظيفة البرمجية trace نبدأ بكتابة أسمها في مكان كتابة الكود كالتالي :



لاحظ أننا اتبعنا trace بقوسين مغلقين (Brackets) لا يحويان أي بيانات وذلك لأنها لم تأخذ أي قيمة لعرضها ، فإذا كان مطلوب منها عرض قيمة نصية كانت أو رقمية هنا يجب إعطاءها هذه القيمة بين القوسين تعالى نجرب

نعطي لها قيمة رقمية بين القوسين مثل 15 مثلا ، ولعمل معاينه للملف Ctrl (+ enter) سنجد القيمة 15 في لوحة output .

أما إذا أعطينا للوظيفة البرمجية trace قيمة نصية ولتكن مثلا أسم شخص ayman هنا يجب وضع الاسم بين علامتى التنصيص هكذا "ayman " ليظهر الاسم ayman في لوحة ال output .



لوحة Compiler errors :

وظيفة Compiler errors هي إظهار الأخطاء الموجودة في الكود ، ومكان وجودها بالضبط في الكود المكتوب مع وصف مبسط لسبب الخطأ .

والأخطاء في البرمجة نوعين :

1- **Sentics errors :** والمقصود بها الأخطاء الإملائية كخطئك في كتابة كلمة trace لو قمنا بزيادة حرف عليها يتغير لونها لأن هذا يعتبر خطأ في كتابتها ولاحظ أن ال Function **الجاهزة تأخذ لون أزرق وهذا من البرنامج** لتميزها .

ولاحظ عند اختبار الملف بعدم ظهور الاسم في لوحة output ولكن هنا تبرز على الفور لوحة Compiler errors لتظهر لى مكان الخطأ في الكود و بمجرد الضغط 2 click على سطر الخطأ يظهر لي مكانه في لوحة الأكشن .

2- **Logical errors :** أي الأخطاء المنطقية حيث لن يعرف بها ال Compiler ، ككتابتك " ahmed" بدلا من "ayman" بالطبع لن يظهر هنا خطأ حيث لن يعرف ال Compiler انك كنت تقصد "ayman" بدلا من "ahmed" .

أو مثلاً إذا قلنا أحسب الكثافة بقسمة الحجم على الكتلة بينما الصح هو الكتلة على الحجم فهنا لن يستطيع ال Compiler معرفة أن هناك خطأ ما ، لآن المشكلة في أنك دخلت معلومات خطأ إذا يجب عليك مراجعة الكود في حاله كانت النتائج خطأ .

الفصل الثاني

المتغيرات والثوابت

Constant &Variable

أولا- المتغيرات Variable

المتغيرات جزء أساسي من أي لغة برمجة ، عند تصميمك لموقع أو لعبة مثلا فأنت تتعامل مع بيانات مثل وقت اللعبة أو عدد المحاولات الناجحة والفاشلة أو عدد الكائنات المكونة للعبة كل هذا وأكثر، بيانات متغيره منها ما هو رقمي ومنها ما هو نصي ، وتم تسمية هذه البيانات بأنها متغيرات Variable لأنها لها صفة القابلية الدائمة للتغير سواء بالزيادة أو النقص ، ولكي تتعامل مع هذه البيانات يجب أولا تخزينها .

كيفية عمل المتغيرات Variable :

سبق وتحدثنا عن لوحة Compiler errors وقلنا أن الـ Compiler وظيفته مراجعة الكود والتأكد من صحته وتكملا لوظيفته فهو يقوم بقراءة الكود إذا كان صحيحاً وتطبيقه على ملف العمل ليخرج العمل كما المطلوب .

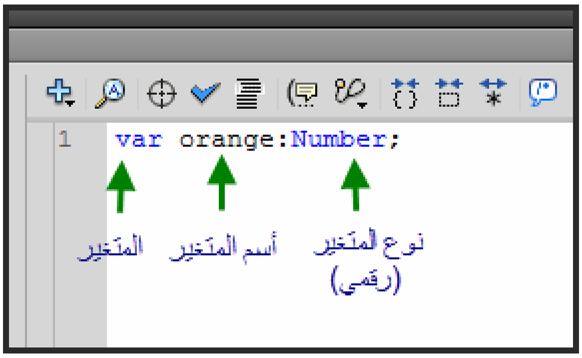
ولذا لعمل Variable يجب أن أطلب من ال Compiler أن يعمل لي متغير وأخبره من خلال الكود باسم المتغير ونوعه حتى يقوم ال Compiler بتنفيذ المطلوب .

أنواع المتغيرات :

هناك أنواع كثيرة من المتغيرات سوف نتعرض لأهما :

- متغير من النوع Number

لعمل **Variable** من النوع Number يتم كتابة الكود كالتالي :



شرح الكود :

1- البداية باستدعاء أو النداء على ال **Variable** من خلال الاختصار **var** ثم مسافة ليظهر لون كلمة **var بالللون الموف بمعنى أنها كلمة محجوزة** في لغة البرمجة .

2- كتابة أسم المتغير هذا الاسم بحسب ما يترآى لي ويفضل أن يعبر عن موضوع المشروع الذي أقوم بتصميمه.

شروط أسم المتغير :

- أن لا يبدأ الاسم برقم ولكن يسمح أن يكون الرقم بداخل الاسم أو بآخره .
- أن لا يكون الاسم محجوز في لغة الأكشن كأن أسمى المتغير **var** مثلا أو **trace** .
- أن يعبر الاسم عن الكائن موضوع التصميم ،

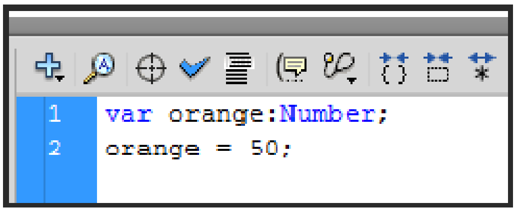
3- نقطتين : (colon) لتحديد نوع المتغير حيث يظهر عدة تلميحات لأنواع مختلفة من المتغيرات يمكنك اختيار من بينها النوع المناسب .

4- نوع المتغير وهنا المتغير من النوع رقمى Number سواء كان الرقم سالب أو موجب أو رقم صحيح أو علامة عشرية ولاحظ بداية الكلمة بحرف capital .

5- انتهاء الكود بفاصلة منقوطة ;

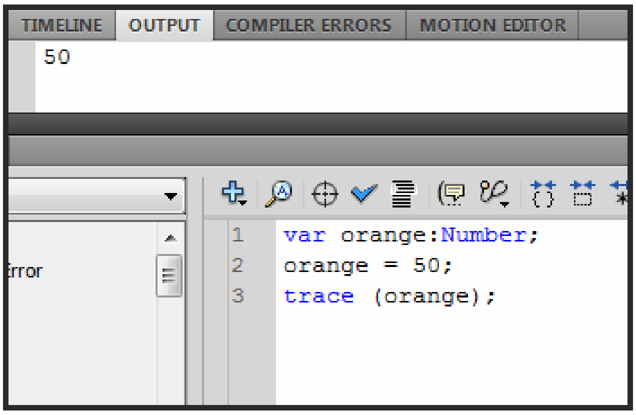
بالكود السابق أعلمت ال **Compiler** بأني أريد عمل متغير وأعلنت عن أسمه كما سبق لكنى لم أحفظ بداخله أي قيمة .

لحفظ قيمة داخل المتغير نكتب الكود التالي :

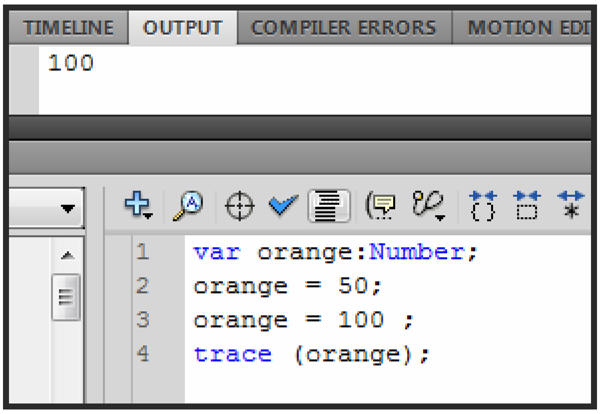


أسم المتغير ثم علامة = ثم الرقم الذي أريد أن أخذنه داخل المتغير و وهنا علامة = في لغة الاكشن سكريبت ليست تساوي التي نعرفها وإنما هي علامة assignment أي ضع القيمة الموجودة يمين العلامة = داخل القيمة الموجودة شمال العلامة = .

وللتأكد فعلا أن المتغير orange يحوى القيمة المعطاة له وهى 50 ، نستخدم الوظيفة البرمجية trace لتظهر نتيجة الاختبار في لوحة output كالتالي :

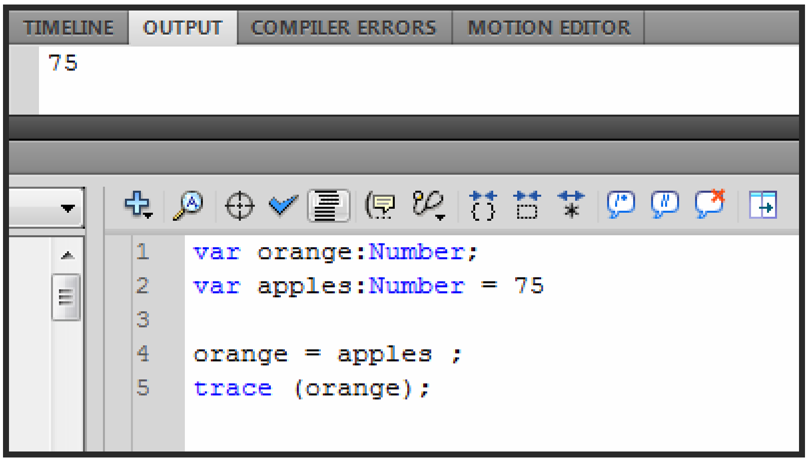


إذا قمنا بإعطاء قيمة جديدة للمتغير orange ولتكن 100 ثم قمنا بالكشف عن القيمة التي بداخل المتغير orange ماذا سنشاهد داخل لوحة ال output ؟ تعالى لنختبر القيمة من خلال trace



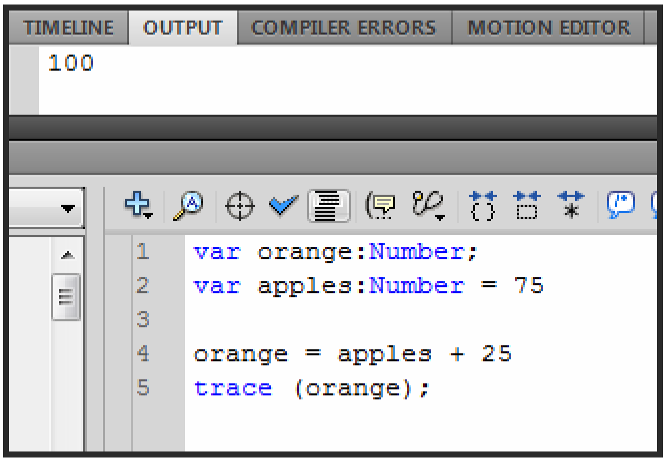
من الصورة السابقة نجد أن ال **Compiler** أخذ القيمة الجديدة للمتغير 100 ومسح القيمة القديمة كما يظهر أمامك في لوحة output .

- ويمكننا الاستدلال عن قيمة متغير بقيمة متغير آخر كالكود التالي :



وفي الكود السابق قمنا بتعريف متغير آخر apples وأعطيناه قيمة 75 ، ثم قمنا بإعطاء المتغير الأول orange قيمة المتغير الثاني apples من خلال الاستعاضة باسم المتغير apples بدلا من قيمته .

- يمكننا القيام ببعض العمليات الحسابية على المتغيرات كأن نجمع أو نطرح أو نقسم أو نضرب قيمة على المتغير كالتالي :



حيث قمنا بجمع القيمة 25 على المتغير apples ، هنا ال **Compiler** قام بجمع القيمة يمين العلامة = و وضعها بداخل المتغير شمال العلامة = .

- **متغير من النوع int :**

وهو متغير من نوع رقمي أيضا ولكنه يستخدم مع المتغيرات التي تأخذ قيمتها أعداد صحيحة سواء سالب أو موجب .

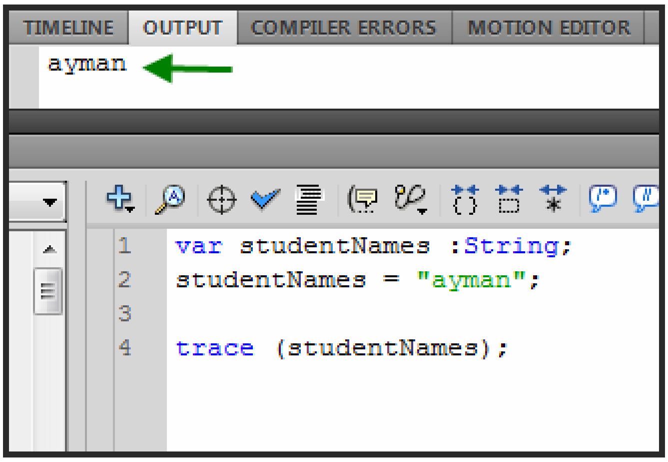
- **متغير من النوع uint :**

وهو متغير من نوع رقمي أيضا ولكنه يستخدم مع المتغيرات التي تأخذ قيمتها أعداد صحيحة موجبة فقط .

- **متغير من النوع String :**

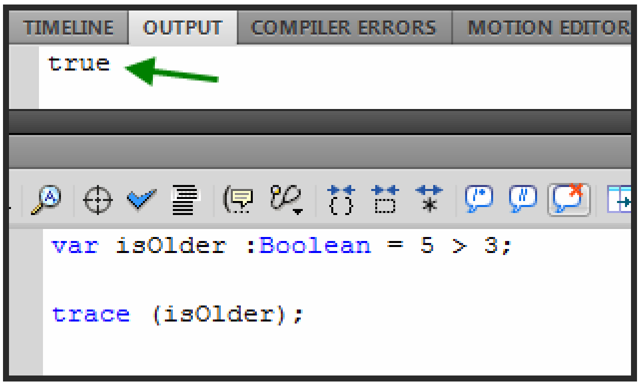
وهو المتغير الذي يحمل قيم نصية ، ويجب وضع القيم النصية بين علامتى تنصيص " "

وكل ما هو بين علامتى التنصيص يعتبر نص حتى لو كتبنا أرقام ، ولذا البرنامج ميز لون القيم النصية بلون أخضر .



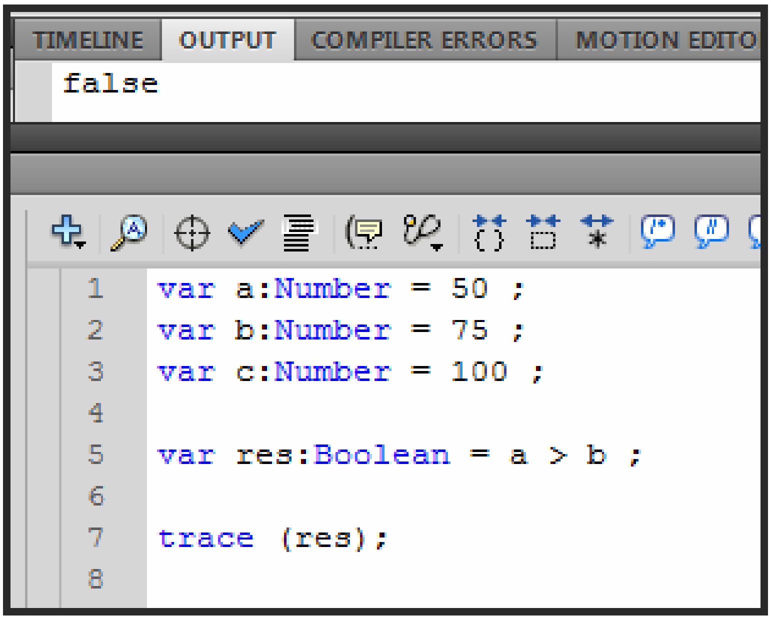
▪ متغير من النوع Boolean

ويختص بالمتغيرات التي نخزن فيها نواتج عمليات منطقية التي لا تحتمل إلا إحدى القيمتين إما true أو false



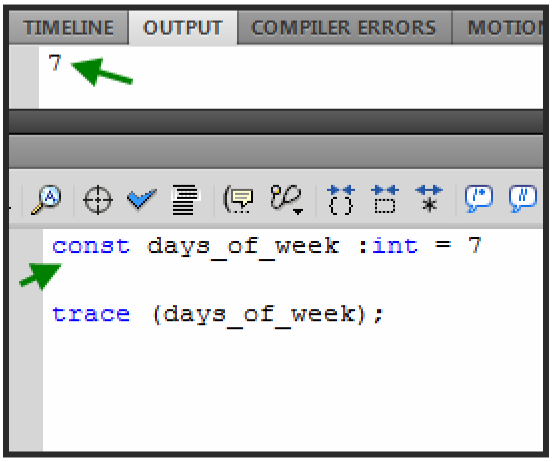
و هنا قمنا بتعريف متغير أسمه isOlder من النوع Boolean وأعطيناه قيمة و هي 3 < 5 و لأن فعلا 5 أكبر من 3 لذا كانت النتيجة المنطقية في لوحة output هي true و كذلك لوجرينا

العكس بأن نسأل هل 3 أكبر من 5 سوف تكون النتيجة المنطقية هي false .



الثوابت Constant

الثوابت هي عكس المتغيرات و طريقة التعامل معها في كتابة الكود هيا نفسها طريقة كتابة الكود في المتغيرات غير أن القيمة التي تعطى للثوابت غير جائز أن تتغير مثل أيام الأسبوع مثلا عددهم 7 ومستحيل يكون عددهم 8 مثلا ، وفرضا لو أعطينا الثوابت قيمة أخرى غير القيمة المفروضة سوف يظهر لنا خطأ هذا الكود في لوحة ال **Compiler** .



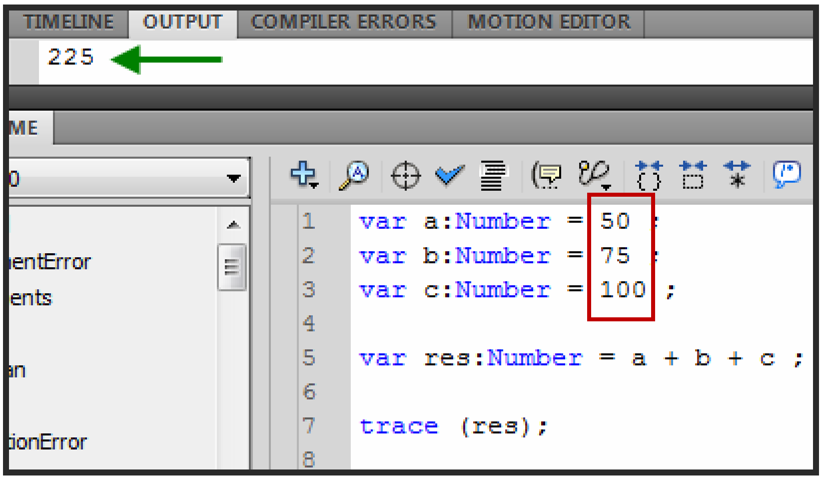
لاحظ في الكود بدلا من كلمه var تظهر كلمة const وهي اختصار لكلمة Constant ، وقمنا بتسمية الثوابت وإعطاءها نوع رقمي صحيح و تخزين بداخلها قيمة رقمية وهي 7 .

الفصل الثالث
العمليات البرمجية
Operator

العمليات: هناك أكثر من نوع للعمليات التي تجرى داخل الاكشن الأسكريبت

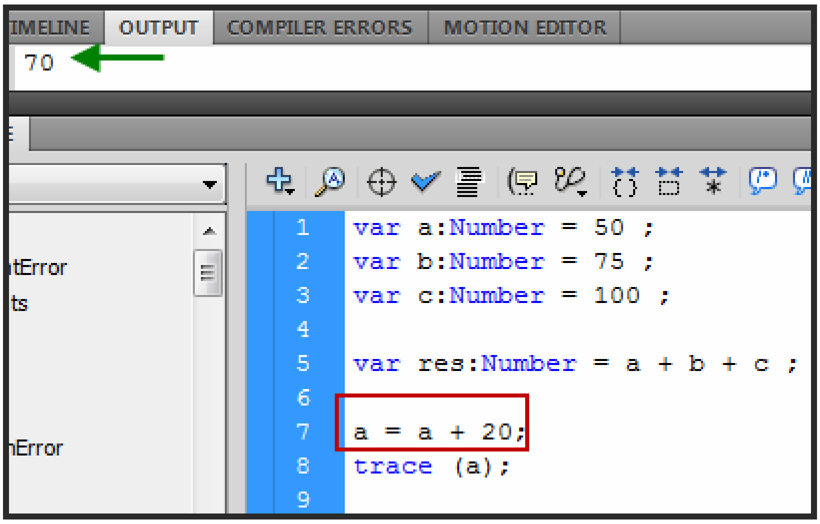
العمليات الحسابية:

العمليات الحسابية المعرفة للجميع وهي الجمع والطرح والضرب والقسمة تتم كالتالي



قمنا في الكود السابق بتعريف عدد من المتغيرات a , b , c وتخزين بداخلها قيم مختلفة 50, 75,100 ثم قمنا بتعريف متغير رابع res لنخزن فيه النتائج الخاصة بالمتغيرات الثلاثة ، وقمنا بعملية جمع لقيم المتغيرات الثلاثة فما كان من ال **Compiler** إلا أن قام بالنظر إلى القيم الخاصة بالمتغيرات الثلاثة ثم جمعهم ووضع نتيجة الجمع في المتغير res شمال علامة = ، وللتأكد كشفنا عن القيمة المختزنة داخل res من خلال trace لتظهر لنا النتيجة 225 في لوحة output

ويمكننا الاستعاضة بدلا من أسم المتغير بقيمة رقمية أيا كانت ، كأن نزود قيمة على أحد المتغيرات كما يظهر في الكود التالي ، حيث قمنا بإضافة 20 على المتغير a



نحتاج غالباً لعملية إضافة قيمة معينة بعد فترة على المتغير وخاصاً في الألعاب لذا فبدلاً من جمع قيمة على المتغير كما في الكود السابق يمكننا الاختصار كالتالي:

a += 20 ; وهي نفسها a = a + 20;

نفس الكلام إذا كنا سنطرح قيمة رقمية معينة من قيمة المتغير فبدلا من علامة + نضع علامة –

a -= 20 ;

ونفس الكلام بالنسبة للضرب a *= 20 ; وكذلك القسمة a /= 20 ;

ويمكن أيضا القيام بعمليات حسابية أخرى مع الكود السابق كأن نقول :

a += 20 * b ;

وهنا يقوم ال **Compiler** بضرب قيمة b وهى 75 في 20 ثم إضافة الناتج وهو 1500 إلى قيمة a لتكون النتيجة 1550

- من ضمن العمليات التي نحتاجها كثيراً وهي زيادة رقم 1 أو إنقاص رقم 1 فيمكننا كتابة ذلك أما بالطرق القديمة المعروفة أو عن طريقة إضافة علامة **++** وينطبق نفس الكلام على إنقاص 1 حيث نضيف علامة **--** .
- ولكن بالطبع لا يوجد علامة ** أو // حيث لا معنى لهذا في العمليات الحسابية لظهور نفس القيمة .

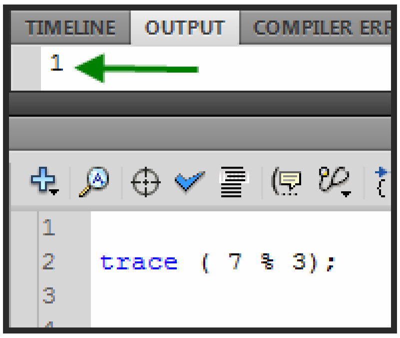
a++ ;	=	a += 1 ;	=	a = a + 1 ;

- ملحوظة : هناك فرق بين أن تكون علامة **++** قبل المتغير أو أن تكون بعد المتغير فإذا كانت بعد المتغير فهذا معناه قم بزيادة 1 ولكن بعد انتهاء السطر البرمجي أي أن هذا يظهر نتيجته على السطر التالي ، أما إذا كانت علامة **++** قبل اسم المتغير فإن الزيادة برقم 1 تأتى قبل نهاية السطر البرمجي وتظهر في النتائج

العملية الحسابية Modular :

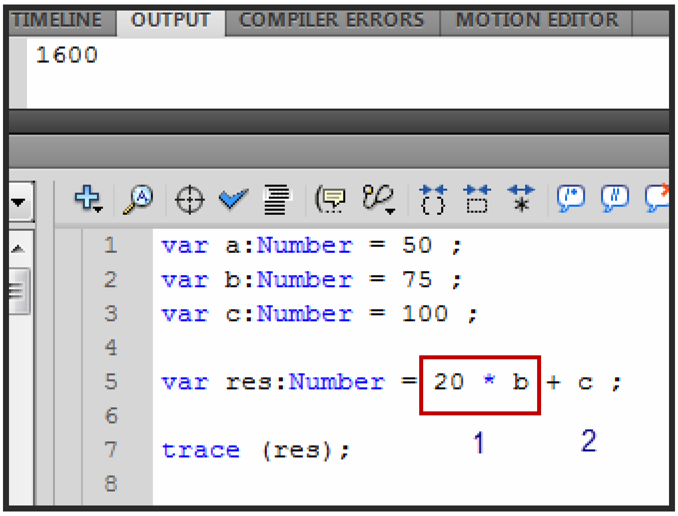
هي نوع من أنواع القسمة ولكنه مختلفة عن القسمة العادية ويرمز لها بعلامة % ، ليكون ناتج العملية رقم صحيح وهو باقي القسمة.

كأن نقسم 7 % 3 لتكون النتيجة هنا = 1 حيث 7 / 3 = 2 والباقي 1



:Operator precedence

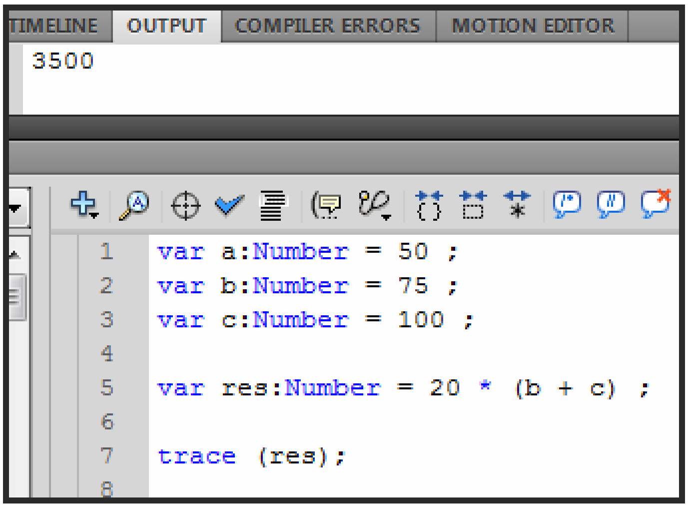
يمكننا القيام بأكثر من عملية حسابية مختلفة في نفس السطر ولكن يتوقف الأمر على أي من العمليات الحسابية ستتم الأول هنا يستخدم ال Compilerمفهوم Operator precedence ويقصد به ترتيب تنفيذ العمليات ومن له الأسبقية في التنفيذ .



وفى مثالنا هذا كان ترتيب العمليات هو ضرب القيمة 20 * b ثم جمع الناتج على c .

وكقاعدة عامة : تنفيذ عمليات الضرب والقسمة يسبق عمليات الجمع والطرح وهناك جدول لترتيب تنفيذ العمليات **Operator precedence** يمكن الرجوع إليه .

ولكن الكشن سكريبت يمتاز بالترتيب والتصنيف ولذا يمكننا الاستغناء عن الجدول من خلال الأقواس التي تنظم وترتب لي أي من العمليات سيتم أولا ، حتى أنك يمكن أن تفرض ترتيب معين كأن يتم جمع الرقمين أولا ثم نضرب الناتج في الرقم الثالث هكذا .ويجب بالطبع قبل القوس المفتوح



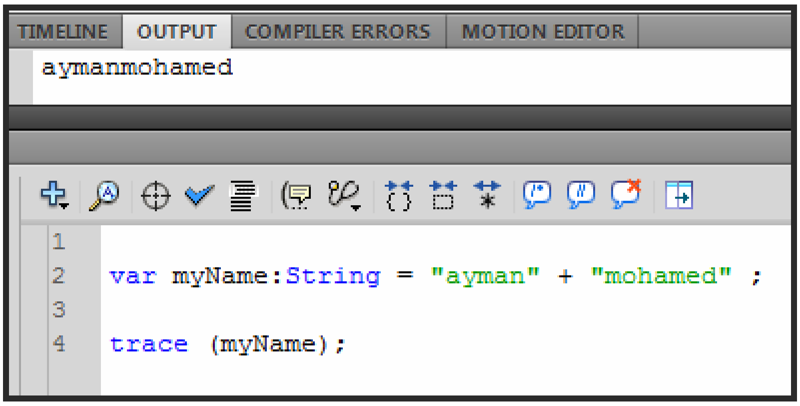
ما سبق كان عمليات حسابية على قيم رقمية أو عمليات حسابية على متغيرات ذات قيم رقمية ، لكن هل يوجد عمليات حسابية على قيم نصية ؟

هنا تنفرد عملية الجمع بهذه الميزة عن غيرها من العمليات ، ولكن كيف يتم هذا و هل هناك جمع لقيم نصية ؟ تعالى لنرى ذلك على المثال التالي :

قمنا بتعريف متغير myName ثم قمنا بتخزين قيمتين نصية وهما "ayman"+ "mohamed" ، وعند اختبار ذلك بـ tarce نجد في لوحة output

القيمتين mohamed ayman بجانب بعض.

ما حدث هنا ليس جمع بمفهوم القيم الرقمية ولكنة يسمى بال concatenation أي تسلسل وربط للقيمتين بجوار بعض وتخزينهم في المتغير myName

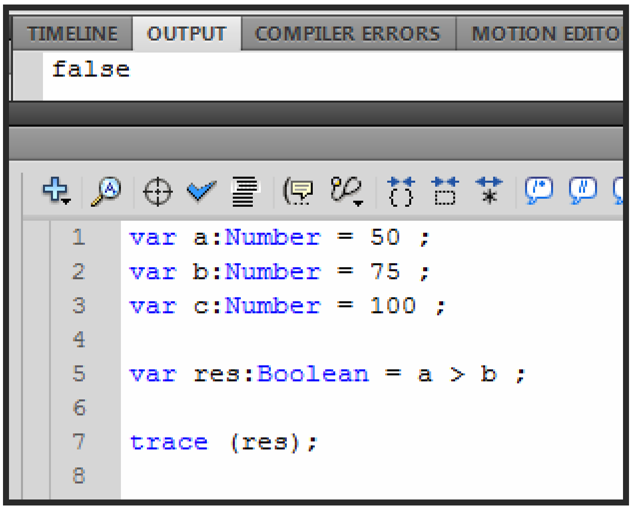


ملحوظة:

في حالتنا هذه للربط بين قيمتين نصيتين عن طريق + إذا كان أحد القيمتين نص والأخر رقم هنا يعامل الرقم أيضا على أنه قيمة نصية حتى لو لم تكن بين علامتى التنصيص.

العمليات المنطقية :

سبق وتحدثنا في الفصل السابق عن نوع من المتغيرات وهو من النوع Boolean ، تعالى لنطلق العمليات الخاصة بهذا النوع .



كما نشاهد هناك عدد من المتغيرات تحمل قيم رقمية مختلفة ، وقمنا بعمل متغير باسم res من النوع Boolean ثم أردنا تخزين نتيجة العملية المنطقية بداخله ، و هنا سألنا : هل قيمة المتغير a أكبر من قيمة المتغير b فكانت النتيجة في ال output بـ false .

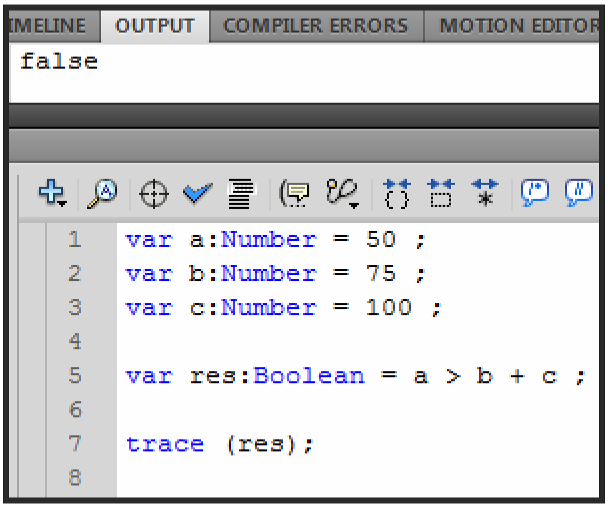
وبالطبع لو كانت a > b كانت النتيجة بـ true .

والعمليات المنطقية تشمل المقارنات التالية:

أكبر من >
أكبر من أو يساوي >=
أصغر من <
أصغر من أو يساوي <=
يساوي == (وهي تختلف عن علامة = التي تعنى assignment)
لا يساوي != (علامة التعجب ! ثم علامة =)

- هل يمكن استخدام العمليات الحسابية في المقارنات المنطقية ؟

بالطبع يمكن ذلك كأن تتم العملية حسابية في أحد طرفى المقارنة أو أن يكون كل طرف يحوى عملية منطقية منفصلة .



في ملاحظة أن العمليات الحسابية تتم أولا ثم يتم المقارنة بين الناتج وبين الطرف الأخر ،

وهنا قمنا بجمع قيمه المتغير c مع قيمة المتغير b ثم تم مقارنة الناتج مع قيمة المتغير a

ويمكننا استخدام الأقواس كناحية تنظيمية .

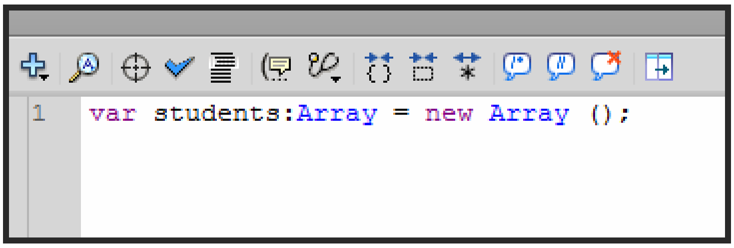
الفصل الرابع
المصفوفات Arrays

المصفوفات Arrays:

وهي عبارة عن مجموعة من البيانات أو العناصر المرتبة والمتشابهة ، كل عنصر فيها له ترتيب معين داخل المجموعة ، كأن نقول مثلا مجموعة كتب arrays books ، في الأكشن سكريبت المصفوفات هي كائن objects له شكل معين وخصائص معينة .

كيفية تعريف Array :

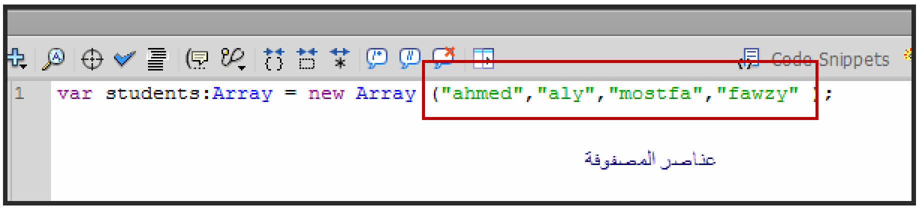
مثل تعريف objects أو Variable :



وبكده بعلم الـ Compiler بوجود Array بالاسم Students لتخزين بداخلة مجموعة من البيانات المتشابهة الخاصة بالطلاب .

طريقة إضافة بيانات أو عناصر المصفوفة :

إضافة العناصر بين القوسين بعد كلمة New Array كما يلي :



وكما يبدو من الكود السابق قمنا بتخزين أربعة بيانات نصية بأسماء الطلاب ، ولو أن عرفنا هذه العناصر على أنها متغيرات لاضطررنا لتعريف كل عنصر

منفصل عن الأخر ، لكن ولأنها عناصر متشابهة فقد تم وضعها في شكل مصفوفة واحدة تحمل جميع أسماء الطلاب ويسهل التعامل معها .

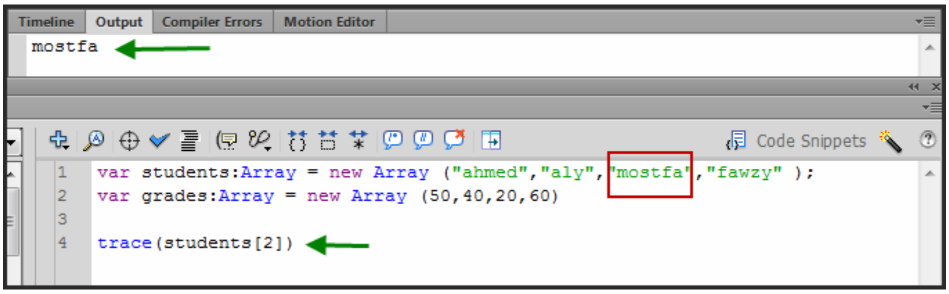
تعتبر الـ Arrays نوع من أنواع الـ objects الجاهزة بخواصها إلا إنها تختلف عنها من حيث أن ال objects كائن واحد له عدة بيانات أو خصائص مختلفة ، بينما ال Arrays بيانات متشابهة لكائن واحد .

وتكملاً لما سبق يمكننا عمل مصفوفة أخرى نضع بها درجات الطلاب لذا قمنا بعمل مصفوفة وخزنا بها قيم رقمية .

خاصية طول المصفوفة (length) :

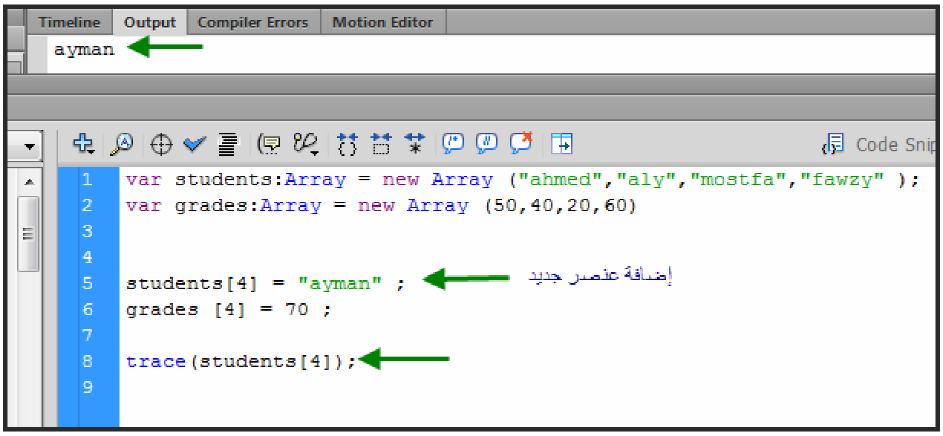
طول المصفوفة هو عدد عناصر المصفوفة فإذا كانت عناصر المصفوفة أربعة عناصر إذا فإن طول المصفوفة هو 4 ، ولأننا في لغة الأكشن سكريبت نبدأ العد من الرقم 0 فإن ترتيب العناصر يبدأ من الرقم 0 ، وفي مثالنا هذا يكون ترتيب الطالب الأول ahmed هو 0 والطالب الثاني ترتيبه 1 وهكذا ، والترتيب في لغة الأكشن هنا يسمى index .

ومن خلال معرفتي لترتيب العنصر استطيع الوصول لقيمته كالتالي :

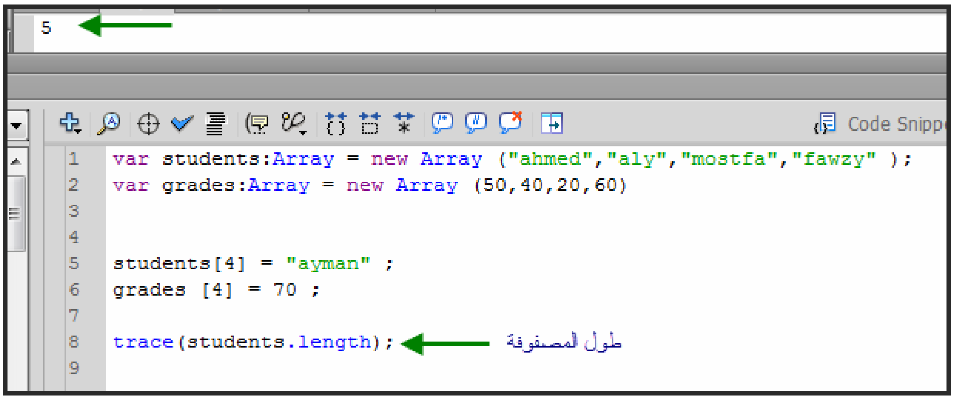


هنا كشفنا بـ trace عن قيمة العنصر الذي رقم الـ index الخاص به هو 2 ، وذلك من خلال كتابة أسم المصفوفة ثم ترتيب العنصر .

فرضاً إني أريد زيادة عنصر جديد على عناصر المصفوفة السابقة ، هذا يتطلب مني كتابة أسم المصفوفة ثم رقم الـ index الخاص بالعنصر الجديد ثم علامة = ثم أسم العنصر الجديد .

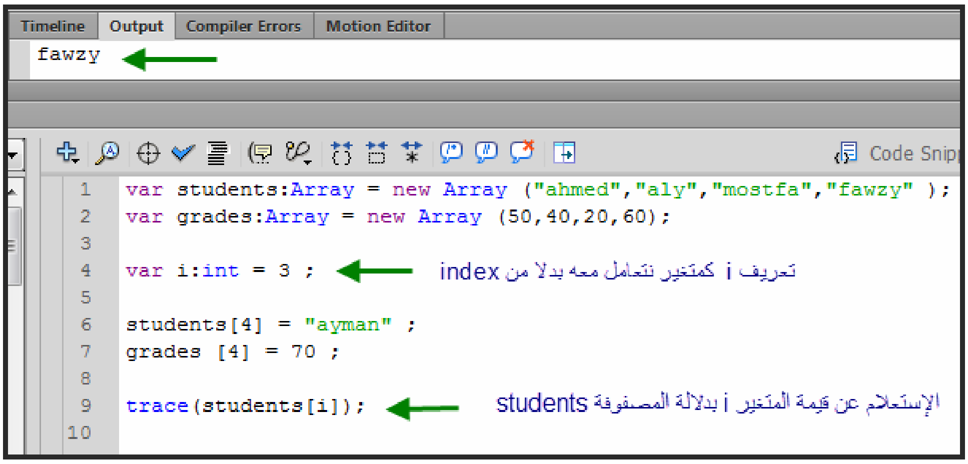


تختص الخاصية **length** بطول المصفوفة أو عدد عناصر المصفوفة كما سبق وتحدثنا ، وكما نفعل مع الـ objects في الكشف عن الخصائص ، بنفس الطريقة نكشف عن الخاصية length .



نكتب أسم المصفوفة ثم نقطة ثم length ليظهر لي عدد العناصر في لوحة output وهم 5 عناصر ، وإذا أضفت عنصر أو حذفت تتغير الخاصية length تلقائيا ويخزن فيها عدد العناصر الفعلية .

ويمكننا بدلا من استخدام قيمة ثابتة نستطيع أن نستخدم متغير نخزن فيه ترتيب العنصر ،لأننا لن نكون على علم تام بعدد العناصر ، وممكن تغيير قيمة المتغير وبالتالي الحصول على قيمة أي عنصر بسهولة .



ولا يشترط لعمل المصفوفة تخزين قيم نصية أو رقمية بل يمكننا تخزين داخل المصفوفة objects كمجموعة من الصور أو الفيديو .

الفصل الخامس
الحلقات التكرارية
Loops

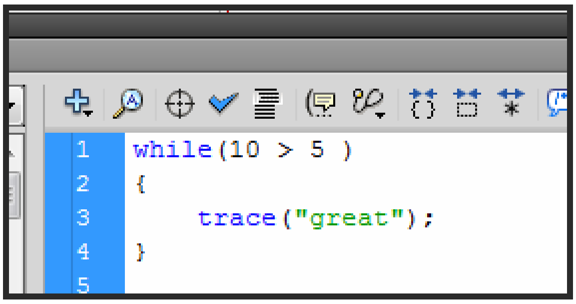
الحلقات التكرارية Loops :

مفهوم معروف لكل لغات البرمجة ، وهو يمكنني من تكرار كود معين أو مجموعة من السطور لعدد معين من المرات لسبب ما ، كأن يكون هناك عدد من الخطوات تستخدم لحساب درجات طالب ما وهي نفسها خطوات حساب الدرجات لباقي الطلاب ، فليس من العملي تكرار كتابة الكود لكل طالب على حدا وخاصاً إذا كان عندي عدد كبير من الطلاب .

Loops تمكنني من تكرار كود معين لعدد معين من المرات أيا كان العدد .

أنواع الـ Loops :

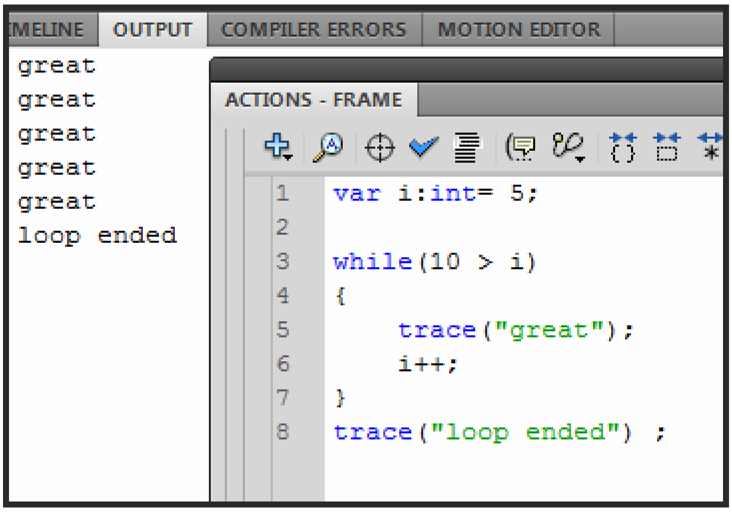
النوع الأول while



بدأنا بـ while بمعنى طالما ثم وضعنا الشرط وهو في المثال 5 < 10 ثم قوس متعرج { ولا ننسى قفله } بعد وضع جسم الـ Loops أو السطر المراد تكراره .

طالما 5 < 10 كرر الكود أو السطر الذي بين القوسين المتعرجين ، ولو قمنا باختبار الكود المكتوب سوف يؤدى ذلك إلى توقف البرنامج عن العمل ، لان عند قراءه الـ Compiler للكود سيجد بالفعل أن الرقم 10 اكبر من

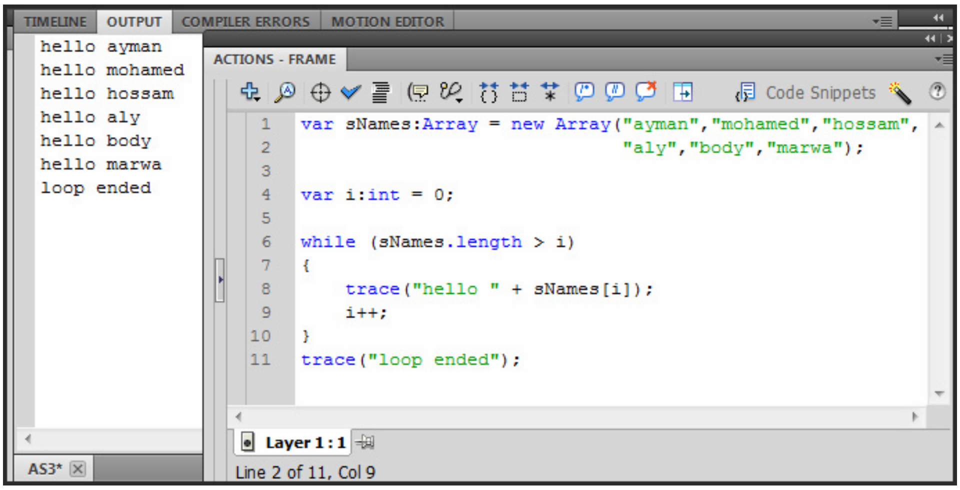
الرقم 5 وبالتالي سوف يتم تنفيذ ما بين القوسين المتعرجين ثم يعود لقراءة الكود من الأول مرة أخرى وبالتالي تكرار ما بين القوسين المتعرجين إلى مالا نهاية . لذا يجب كسر الكود لمنع عدم تكراره عند نقطه معينة .



قمنا بعمل متغير i كعداد نخزن بداخله قيمة رقمية 5 ثم نقارن القيمة الرقمية 10 بالمتغير i ، إلى هذا الحد الأمر لا يختلف عن ما سبق حيث ستظل 10 > i ، لذا قمنا بعد سطر الـ Loops بزيادة ++i أي زيادة 1 على المتغير i كل مرة ، وبهذا نكسر تكرار الكود حيث سيصل المتغير i بعد 5 مرات إلى الرقم 10 وهنا لا يصبح i > 10 فنخرج عن تكرار جسم الـ Loops وننتقل إلى قراءة باقي الكود .

وبهذا استطعت تكرار الكود إلى 5 مرات دون الحاجة إلى كتابته لـ 5 مرات .

عادةً ما نحتاج لتكرار نفس السطور ولكن على قيم مختلفة كالتالي :

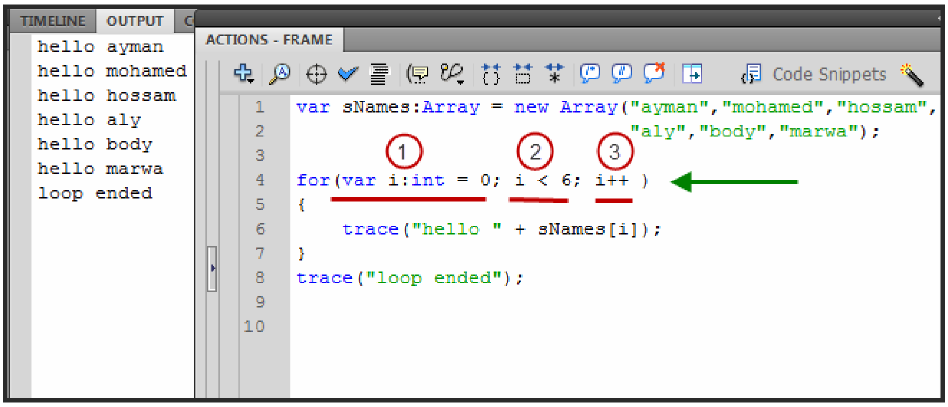


في الكود السابق قمنا بعمل مصفوفة لعدد من الأسماء ثم المقارنة بين طول المصفوفة و الرقم المخزن في المتغير i ، بمعنى طالما طول المصفوفة length أكبر من الرقم المخزن في المتغير كرر الكود لكل أسم في المصفوفة على حدا ، حيث تتغير قيمة i وتزيد كل مره 1 .

النوع الثاني: For

هي شبيهة بالـ while من حيث الفكرة ، ولكن تختلف في طريقة كتابة الكود فيما بين القوسين، حيث يكتب 3 أشياء :

- تعريف متغير كعداد .
- الشرط (الذي يجب اختبار تنفيذه قبل النظر إلى التغير الذي يحدث لكسر التكرار).
- التغير الذي يحدث على المتغير المستخدم كعداد لكسر تكرار الكود.



ثم بين القوسين المتعرجين { } نكتب سطور الكود المراد تنفيذها وتكرارها إلى أن يكسر الشرط فنخرج من الـ Loop وننفذ السطور خارجها .

ولاحظ أنه : بنسبة كبيرة سنستخدم الـ Loops مع المصفوفات في أغلب مشروعاتنا .

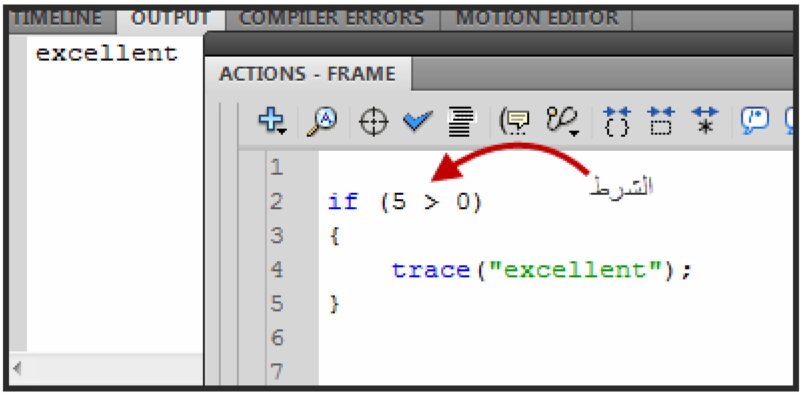
الفصــل السادس
التنفيذ الشرطي
Conditionals

التنفيذ الشرطي Conditionals :

نلاحظ من دراستنا للفصول السابقة أن السطور المكتوبة من أكواد يقوم الفلاش تنفيذها بتتابع ترتيبها ، ولكن في فصلنا هذا سوف نكتب سطور لا تنفذ إلا بتنفيذ شرط معين ، حيث يمكنني الـ Conditionals من تحقق ذلك ، و هنا سنخبر الـ Compiler بأن ينفذ السطور المكتوبة فقط في حالة الشرط المطلوب .

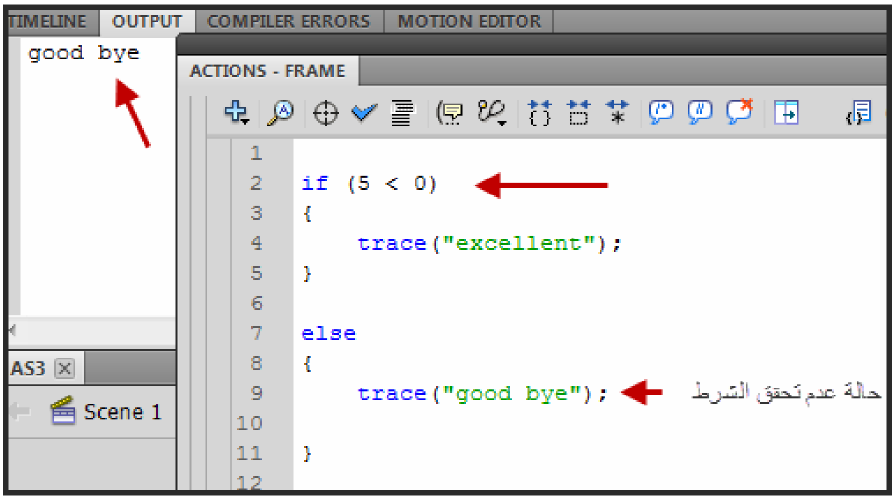
أنواع الـ Conditionals :

النوع الأول – if statement



كتبنا if ثم فتحنا قوسين لوضع الشرط المراد تحقيقه بينهم ، ثم قمنا بفتح قوسين متعرجين لوضع السطر المراد تحقيقه في حالة تنفيذ الشرط أما في عدم تحقق الشرط يتجاهل الـ Compiler ما بين القوسين المتعرجين لتنفيذ باقي الكود .

يتضح من الشرح السابق للكود تنفيذ السطر الذي بين القوسين المتعرجين في حالة تحقق الشرط ، أما في حالة عدم تحقق الشرط ماذا نفعل ؟



في الحالة السابقة لم يتحقق الشرط لذا لجأنا إلى else وهنا ننفذ ما بين القوسين المتعرجين بعد else .

نحب قبل الدخول في الجزئية التالية من درس التنفيذ الشرطي أن نتعرض بشئ من التفصيل لموضوع قد سبق وتحدثنا عنه في درس العمليات البرمجية ألا و هو العمليات المنطقية Logical Operators.

العمليات المنطقية : هي عمليات تتم بين قيم منطقية وناتجها قيم منطقية أما true أو False

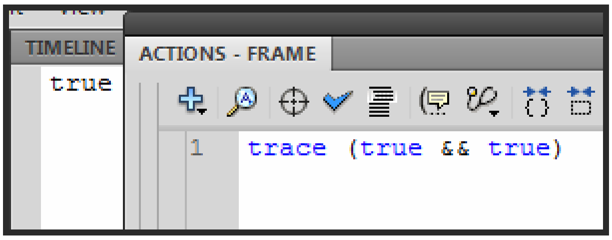
أنواع العمليات المنطقية :

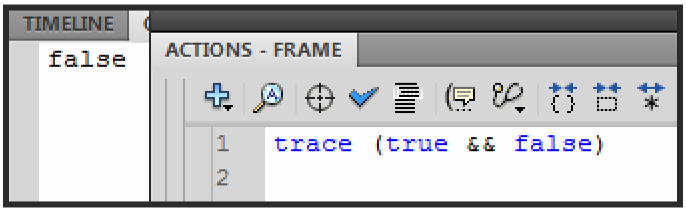
هناك ثلاث أنواع من العمليات المنطقية:

1- && بمعنى و (and) وتأتي بالضغط مرتين على (shift + 7) .

إذا كان طرفي العملية المنطقية تحمل قيمة من النوع true كان الناتج true .

أما إذا كان طرفي العملية المنطقية أحدهما من النوع true والأخرى من النوع false أو كانت القيمتين false كان الناتج false.

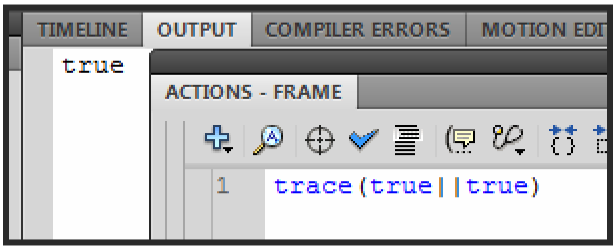


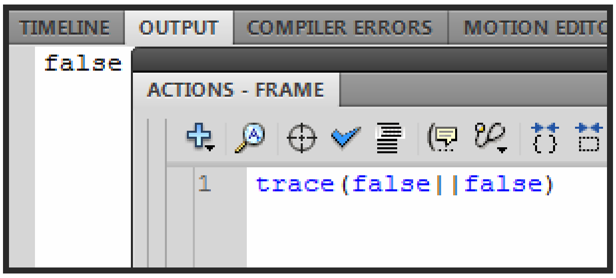


2- || بمعنى أو (or) وتأتي بالضغط مرتين على (\ + shift).

إذا كان طرفي العملية المنطقية أو أحدهما تحمل قيمة من النوع true كان الناتج true .

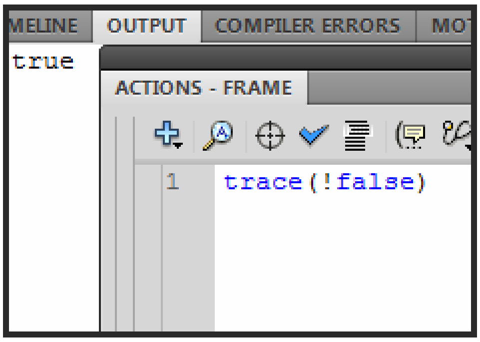
أما إذا كان طرفي العملية المنطقية تحمل القيمة false كان الناتج false.





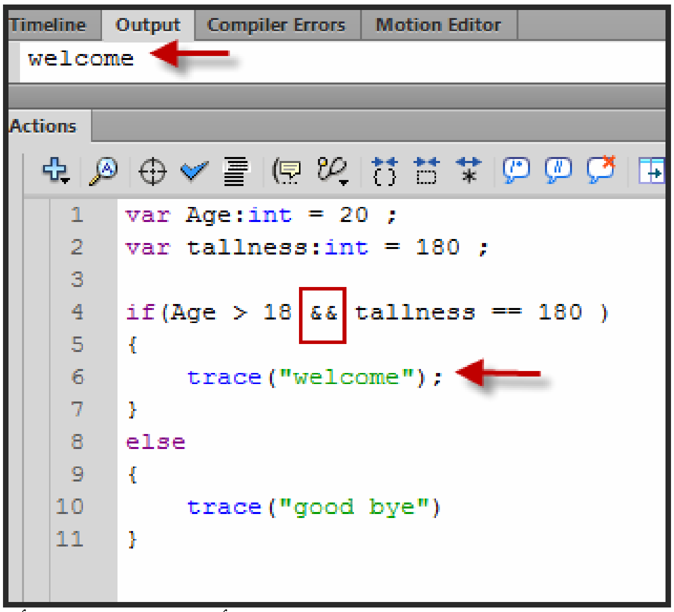
3- ! بمعنى not (لا) وتأتي بالضغط على (علامة التعجب + shift) .

تستخدم مع قيمة منطقية واحدة لعكس القيمة المنطقية هذه كأن تقول !false ليكون true .



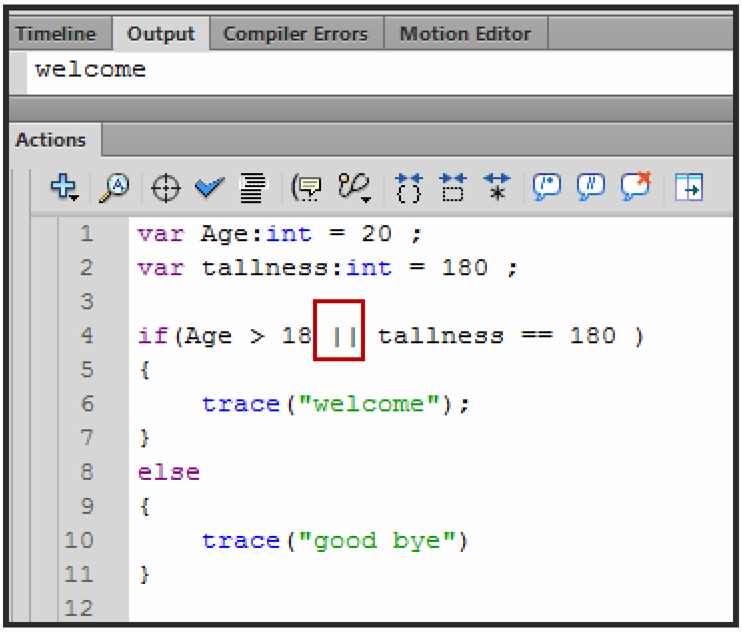
- في كثير من الأحيان لا يكون عندنا شرط واحد لاختباره ، بل عدة شروط نريد اختبارها ، على الأقل شرط واحد منهم يتحقق لتنفيذ الكود ، هنا سوف نستخدم العمليات المنطقية Logical Operators .

مثال : فرضاً إننا نقوم بعمل برنامج لاختبار مدى صلاحية الطلبة المرشحين لإحدى الكليات العسكرية



قمنا بتعريف متغيرين لتخزين البيانات بداخلهما أحدهما للعمر والأخر للطول ، ثم سألنا ب if لو كان السن أكبر من 18 والطول كان نفس الطول المخزن لدينا 180 ، إذا نفذ الشرط ، غير ذلك else نفذ السطر الآخر .

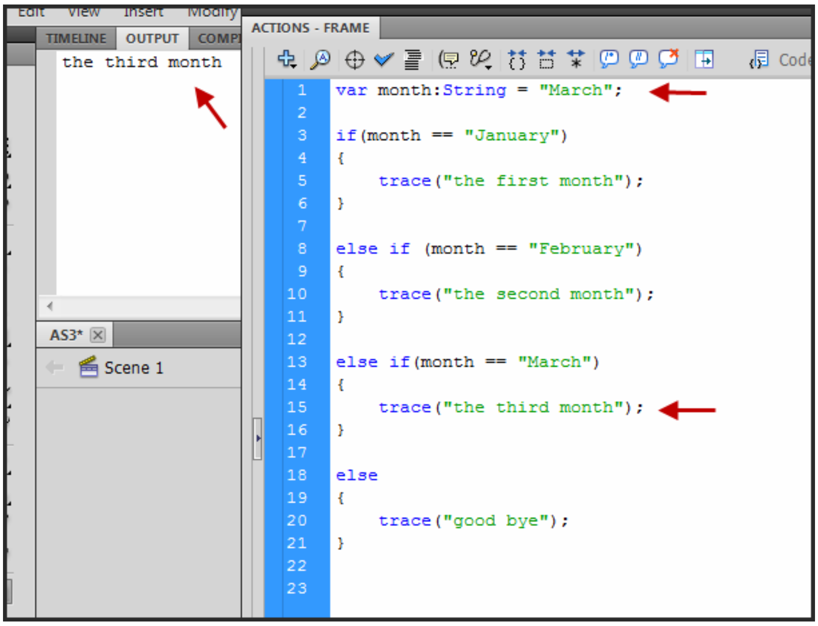
أما في حالة استخدام || (or) بدلا من && ، فهذا معناه في حالة تحقق أحد الشرطيين نفذ الأمر غير ذلك فنفذ الشرط الأخر .



: else if

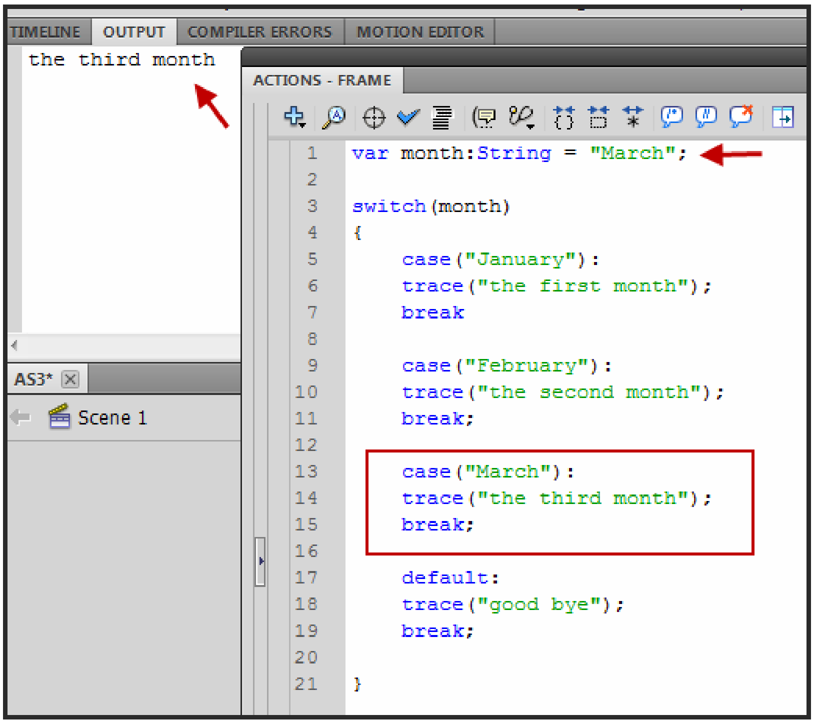
تستخدم في حالة اختبار قيم مختلفة لمتغير ما .

في المثال التالي إذا كانت قيمة المتغير month تساوي القيمة المخزنة January إذا نفذ الشرط الذي بين القوسين المتعرجين ، غير ذلك else if إذا كانت قيمة المتغير month تساوي القيمة February نفذ السطر التالي وهكذا ، أما إذا كانت قيمة المتغير لا تحمل أي من القيم السابق ذكرها هنا نستخدم else فقط لتنفيذ السطر الأخير على باقي الشهور التي لم يأتي ذكرها .



النوع الثاني - switch statement

تشبه الحالة السابقة if من حيث الفكرة والنتائج وتستطيع استخدام أي من الطريقتين إلا إنها تختلف في شكل الكود كالتالي:



عرفنا متغير month ثم نخزن بداخلة القيمة التي تختبر عليها باقي القيم ، ثم نكتب switch وبين القوسين نضع أسم المتغير ، ثم فتح قوسين متعرجين لنضع بينهم جميع الحالات المراد اختبارها :

في حالة القيمة المخزنة داخل المتغير كانت هكذا case("January"):

السطور المراد تنفيذها في حالة تنفيذ الشرط trace("the first month");

كسر الشرط أي الخروج من اختبار باقي الكود في حالة تنفيذ الشرط break;

ثم نكتب حالة ثانية و هكذا ...

إما باقي القيم التي لم يتم ذكرها فينطبق عليها الكود التالي:

```
default:
trace("good bye");
break;
```

الفصل السابع
الوظائف البرمجية
Functions

: Functions

هي وظيفة برمجية عبارة عن مجموعة من سطور الكود لتنفيذ وظيفة معينة تحدد أسمها و مهامها و أهمية الـ Functions في شيئين :

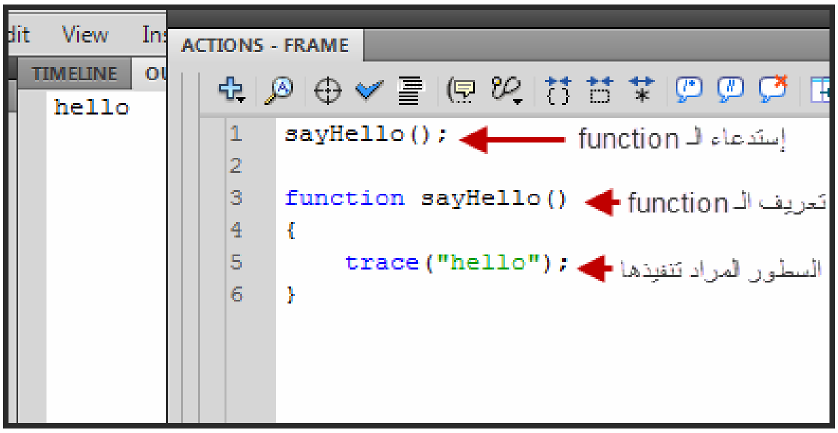
- تقسيم التطبيق الذي تقوم ببناءة إلى مجموعة من الوظائف مما يقلل من الأخطاء ويسهل
في تطوير التطبيق بسهولة .
- توفير كتابة وتكرار الكود في أكثر من مكان الذي نحتاج فيه الكود ،لأننا نكتب الكود مرة واحدة ونعطيه أسم وعند احتياجنا له في مكان آخر نستدعية باسمة السابق .

وكلما كانت قدرتك أكبر في تقسم تطبيقك الذي تبنية إلى مجموعة من الوظائف المنفصلة التي تجتمع مع بعضها لتفعيل التطبيق على أحسن وجه ، كل ما دل ذلك على انك مبرمج محترف وكلما أعطاك من إمكانيات لتطبيق برامج ومشاريع والعاب أكبر وكلما كان الكود الذي تقوم ببناءة سهل الفهم والتعديل .

وهناك العديد من الـ Functions الجاهزة في البرنامج مثل , trace, play , stop , gotoAndPlay , gotoAndStop وغيرها كثير ، ولكن ما سوف نتعلمه الآن هو كيفية عمل Functions خاصة بي وتقسيم الكود إلى مجموعة من الوظائف .

كيفية عمل Functions :

يجب أولا تعريف Functions وأعلام الـ Compiler بذلك ، وبالطبع نقوم بتسميتها ويفضل أن يدل الاسم على الوظيفة الخاصة بها في المشروع ، ثم فتح قوسين متعرجين نضع بينهما السطور المراد تنفيذها في حالة استدعاء هذه الوظيفة .



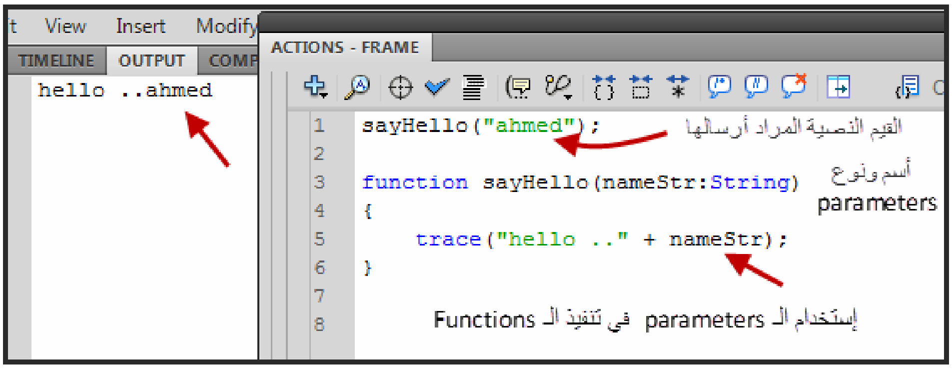
كيفية استدعاء Functions :

من خلال كتابة أسمها متبوع بقوسين مغلقين بدون أي قيمة ، ويمكننا كتابة واستدعاء الـ Functions قبل تعريفها ، و هنا تختلف الـ Functions عن المتغيرات حيث أن الـ Compiler يقوم أولا عند مراجعة الكود بالبحث عن تعريف الـ Functions، لذا لو سبق استدعاء الـ Functions تعريفها فهذا لن يؤثر على صحة الكود .

- من مميزات الـ Functions هي إمكانية تنفيذ نفس السطور على قيم مختلفة من خلال إرسالي لها قيم معينة تنفذ عليها السطور و هذه القيم سواء كانت متغيرات أو قيم ثابتة فهي تسمى parameters .

مثال :

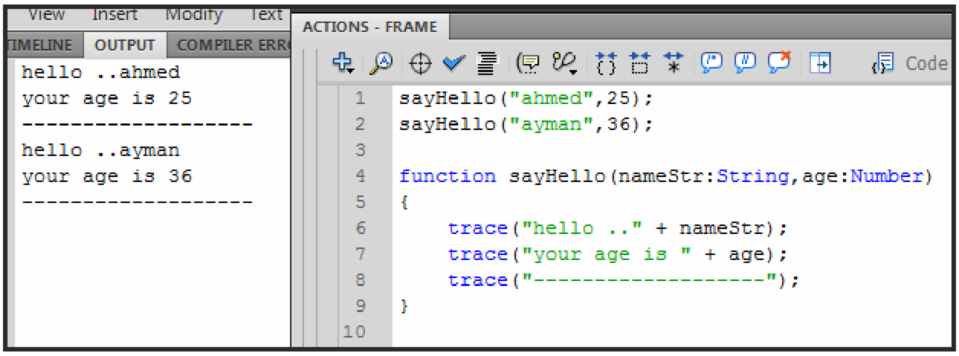
- إذا أردت إعطاء الـ Functions - وهي في مثالنا هذا باسم sayHello - قيمة نصية معينة ننفذ عليها السطور ، يجب أعلام الـ Compiler بأن الـ Functions تستقبل parameters ويتم هذا أثناء تعريفي للـ Functions .
- في مرحلة تعريف الـ Functions نكتب بين قوسين الـ Functions أسم الـ parameters التي يستقبل القيمة النصية المراد إرسالها له ثم نحدد نوعه ولأننا نريد تخزين قيمة نصية داخلة لذا فأن نوعه يكون من النوع string .
- نستخدم القيمة النصية من خلال كتابة أسم الـ parameters في سطور تنفيذ الـ Functions .
- تأتي مرحلة استدعاء الـ Functions من خلال كتابة أسم الـ Functions وبين القوسين نضع القيمة النصية المراد إرسالها .



للملاحظة :

أستطيع استخدام نفس الـ Functions مع قيم نصية مختلفة. كما يمكنني إرسال للـ Functions أكثر من parameters وبالطبع

يجب هنا أعلاء الـ Compiler بأن الـ Functions تستقبل أكثر من parameters ، ونقوم بإضافة parameters جديد يحمل قيمة رقمية كالمثال التالي :



* مثلما أستطيع أن أرسل للـ Functions قيمة أستطيع أيضاً أن أستقبل منها قيمة معينة كنتيجة عملية حسابية مثلا ، تعالى للمثال التالي :

في مثالنا هذا:

- قمنا بتعريف مصفوفة لأسماء اللاعبين .
- ثم تعريف مصفوفة لنتائج اللاعبين .
- ثم تعريف متغير نخزن فية أعلى قيمة ممكنة لنتائج اللاعبين .
- تأتي بعد هذا مرحلة إنشاء Functions أرسل لة index الخاص بلاعب معين فيرد عليا

بقيمة ال score الذي أحرزه اللاعب بنسبة مئوية .

و هنا نكتب بين قوسين الـ Functions أسم المتغير (i:int) ونخزن فية قيمة رقمية .

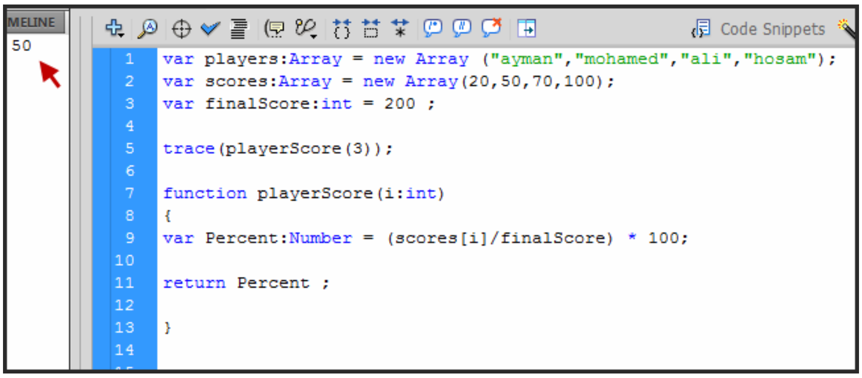
- بعد ذلك وبين القوسين المتعرجين نقوم بعمل العمليات الحسابية الخاصة بالـ Functions فنعرف متغير نخزن بداخلة العملية الحسابية الخاصة بحساب نتائج اللاعب بنسبة مئوية (قسمة نتائج اللاعب على النتيجة الكلية مضروبة في المائة)

var Percent:Number = (scores[i]/finalScore) * 100؛

- وأخيراً نكتب return متبوعة باسم المتغير الحسابي السابق حتى يعود علي بنسبة اللاعب المئوية بمجرد استدعاء الـ Functions بعد إعطاءها index اللاعب .

وللملاحظة تعتبر return أخر سطر في الـ Functions ولا ينفذ بعدها أي سطر .

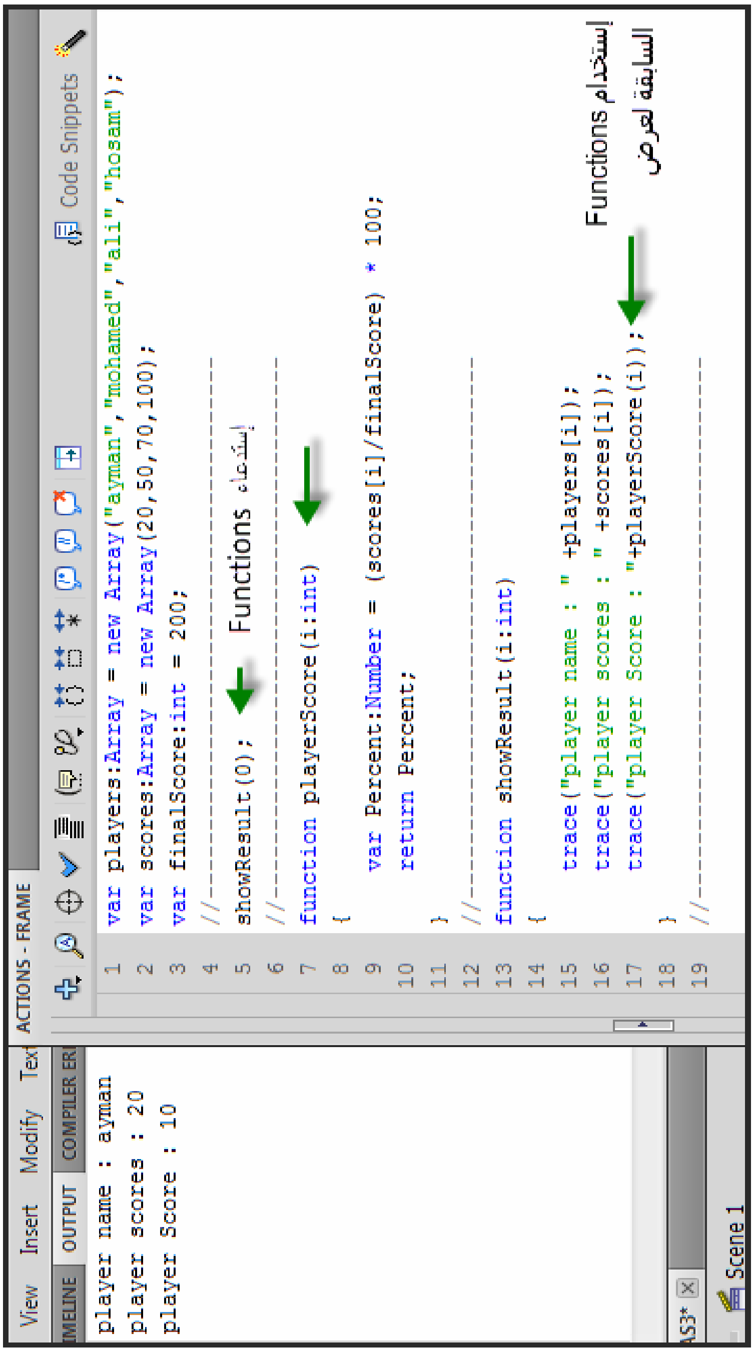
- لاستدعاء الـ Functions نكتب أسمها وبين القوسين رقم الـ index الخاص باللاعب ولكننا هنا نستخدم trace لاختبار القيمة في لوحة الـ output .



وهنا يظهر فائدة الـ Functions في إنك تقوم بمجموعة من العمليات الحسابية مرة واحدة ثم استخدام هذه العمليات مع عدد كبير من الطلاب بمجرد استدعاء الـ Functions .

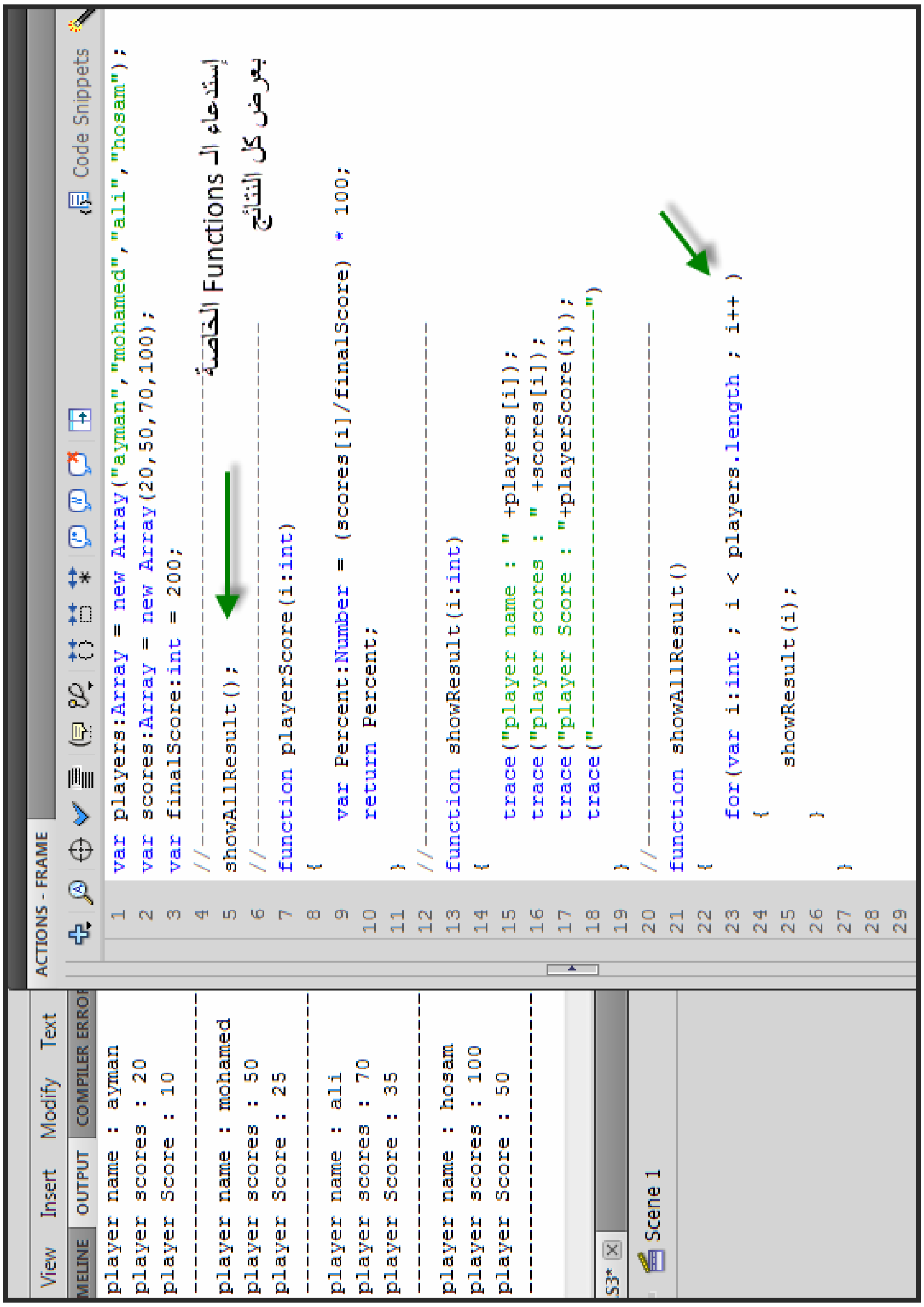
يمكننا استخدام Functions داخل Functions أخرى ، كأن أنشأ Functions لعرض النتائج الخاصة بالطلاب ، وربما أحتاج لعملية حسابية أخرى داخل العملية الحسابية الخاصة بعرض النتائج ، هنا أنشأ Functions أخرى وبها عملية حسابية ما ، ثم استخدامها داخل ال Functions الخاصة بعرض نتائج الطلاب .

تعالى للمثال التالي :


ACTIONS - FRAME
Code Snippets
var players:Array = new Array("ayman","mohamed","ali","hosam");
var scores:Array = new Array(20,50,70,100);
var finalScore:int = 200;
//--
showResult(0);
إستدعاء Functions
//--
function playerScore(i:int)
{
var Percent:Number = (scores[i]/finalScore) * 100;
return Percent;
}
//--
function showResult(i:int)
{
trace("player name : " +players[i]);
trace("player scores : " +scores[i]);
trace("player Score : "+playerScore(i));
}
//--
إستخدام Functions السابقة لعرض
OUTPUT
COMPILER ERRORS
player name : ayman
player scores : 20
player Score : 10
Scene 1

- تكملاً على المثال السابق والخاص بعمل Functions بعرض نتائج اللاعبين بنسبة مئوية .
- قمنا بعمل Functions أخرى لإظهار النتائج الخاصة باللاعبين(showResult) وكان الـ parameters الخاص بها رقمي وأسمه (i) وهو نفس أسم parameters الخاص بالـ Functions المسئولة عن عرض النتائج ، وبرغم تكرار أسم parameters فإن هذا لا يؤثر على صحة الكود لان كل أسم خاص بـ Functions مختلفة .
- وبداخل القوسين المتعرجين نضع ما نريد تحقيقه من عمليات حسابية وعرض للنتائج فكانت أول عملية خاصة باسم اللاعب حسب رقم الـ index ، والثانية خاصة بعرض الدرجة التي حصل عليها اللاعب ، والعملية الثالثة خاصة بعرض نتيجة اللاعب في شكل نسبة مئوية وهنا نحتاج للـFunctions السابقة والتى تمت بداخلها أجراء مثل تلك العملية الحسابية لذا نقوم باستخدامها عن طريق كتابة أسم الـ Functions وبين القوسين نضع أسم الـ parameters وهو هنا (i) لعرض النسبة حسب رقم الـ index للاعب .
- وأخيراً تأتى مرحلة استدعاء الـ Functions متبوعة بقوسين وبداخلهم رقم الـ index الخاص باللاعب المراد عرض نتائجه .

الكود السابق خاص بعرض النتائج اللاعبين ولكن يجب أن تحدد عند استدعاء الـ Functions رقم الـ index الخاص باللاعب ، ولكن فرضاً إننا نريد عرض نتائج اللاعبين مرة واحدة إجمالا ودون تحديد لاعب بعينه ، هنا سنقوم بعمل Functions جديدة كالتالي :


ACTIONS - FRAME
Code Snippets
var players:Array = new Array("ayman","mohamed","ali","hosam");
var scores:Array = new Array(20,50,70,100);
var finalScore:int = 200;
//---
showAllResult();
//---
function playerScore(i:int)
{
var Percent:Number = (scores[i]/finalScore) * 100;
return Percent;
}
//---
function showResult(i:int)
{
trace("player name : " +players[i]);
trace("player scores : " +scores[i]);
trace("player Score : "+playerScore(i));
trace("-----------------------------------");
}
//---
function showAllResult()
{
for(var i:int ; i < players.length ; i++)
{
showResult(i);
}
}
إستدعاء الـ Functions الخاصة بعرض كل النتائج
View Insert Modify Text
MELINE OUTPUT COMPILER ERROR
player name : ayman
player scores : 20
player Score : 10
player name : mohamed
player scores : 50
player Score : 25
player name : ali
player scores : 70
player Score : 35
player name : hosam
player scores : 100
player Score : 50
Scene 1

الـ Functions الجديدة متبوعة بقوسين بدون تحديد أي قيم لأنها لكل اللاعبين بدون تحديد ، ثم قوسين متعرجين وبداخلهم نضع Loop من النوع For (راجع درس الحلقات التكرارية) لنختبر فيها درجات اللاعبين ، فقمنا بتعريف متغير كعداد ، ثم نقارن ما بين رقم الـ index للاعب و طول المصفوفة ككل ، ثم التغير الذي يحدث على المتغير المستخدم كعداد لكسر تكرار الكود

بعد ذلك يمكننا استخدام الـ Functions (showAllResult) والخاصة بعرض جميع نتائج اللاعبين حتى يتم عرض نتيجة كل اللاعبين أجمالا حتى ينكسر تكرار الكود .

وأخيراً يتم استدعاء الـ Functions

```
showAllResult();
```

واعتمادا على نتائج اللاعب ممكن نخرج لة عبارة معينة مثل إذا كانت نتائج اللاعب أكثر من % 40 نكتب له "good" أما إذا كانت النتائج أقل من 40 % نكتب له " try again " ، إذا هنا نستخدم قاعدة if مع الـ Functions الخاصة بالنسبة المئوية وهي (playerScore)

```
function playerScore(i:int)
{
    var Percent:Number = (scores[i]/finalScore) * 100;
    return Percent;
}
//----------------------------------------
function showResult(i:int)
{
    trace("player name : " +players[i]);
    trace("player scores : " +scores[i]);
    trace("player Score : "+playerScore(i));

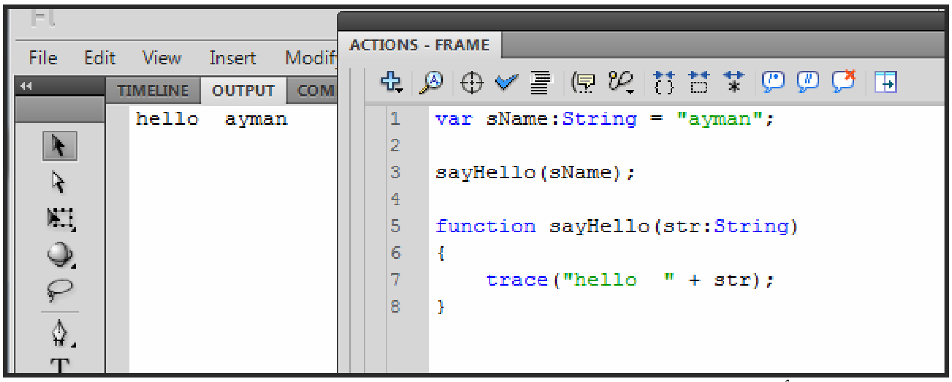
    if(playerScore(i) > 40)
    {
        trace("good");
    }
    else if(playerScore(i) < 40)
    {
        trace ("try again");
    }

    trace("--------------------------------")
}
//----------------------------------------
```

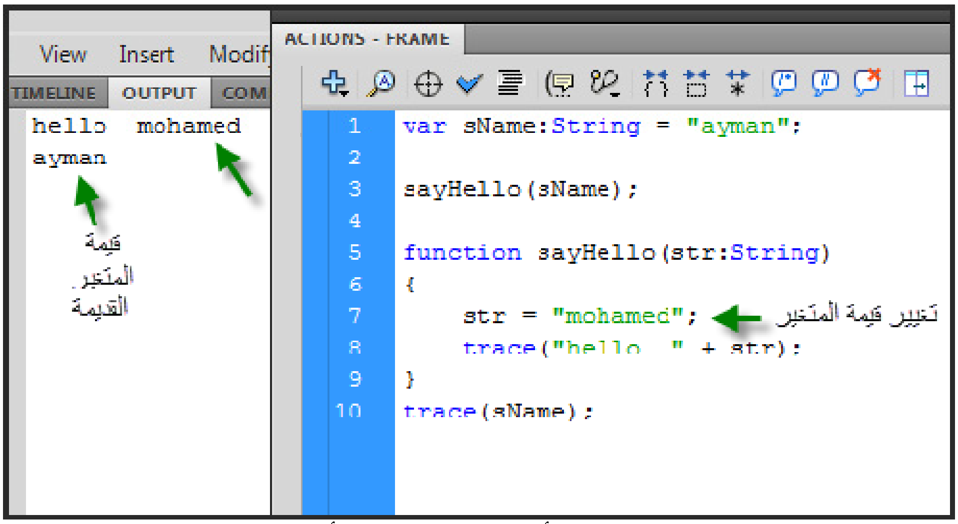
إرسال قيمة معينة للـ Functions هذا يسمى passing و هو نوعان :

النوع الأول : Passing by value :

أي إرسال القيمة المخزنة بداخل المتغير



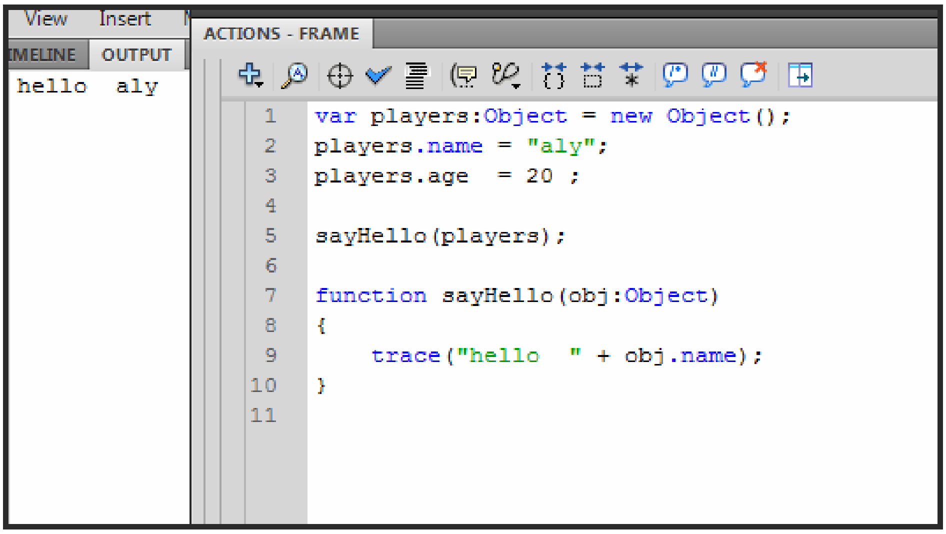
في مثالنا هذا أرسلنا للـ Functions القيمة المخزنة في المتغير وليس المتغير نفسه بدليل عندما قمنا بتغيير قيمة المتغير داخل الـ Functions تغيرت القيمة ولكن المتغير الأصلي مازال يحتفظ بقيمته القديمة



وهذا يحدث تلقائياً مع المتغيرات بأنواعها حيث إننا أرسلنا قيمة وليس المتغير نفسه .

ولكن الأمر مختلف مع الـ object وهذا هو النوع الثاني من الـ passing .

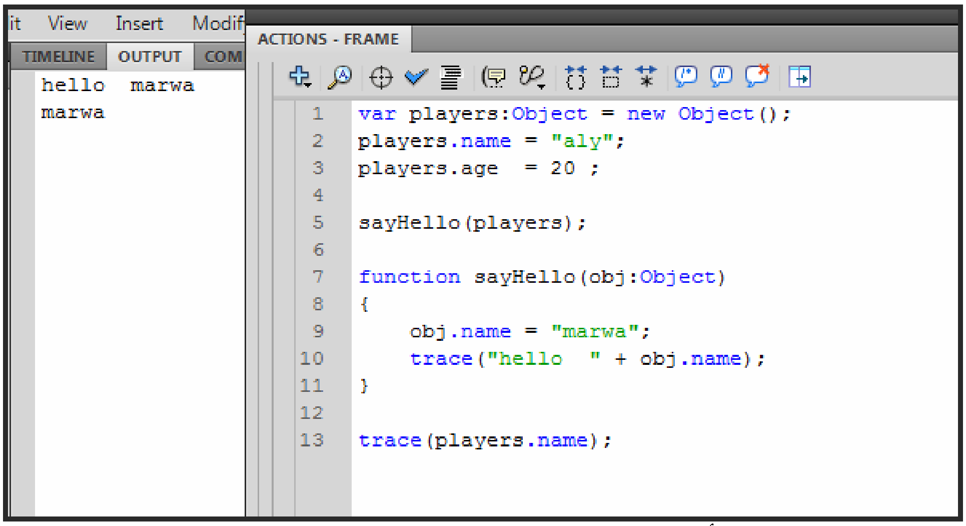
النوع الثاني: passing by reference :



في مثالنا هذا قمنا بتعريف object ثم أنشأنا Functions وكان الـ parameters من النوع

object ثم أرسلنا للـ Functions أسم object لينفذ العمليات علية ، هنا الـ passing يسمى

passing by reference بمعنى إني أرسل لة الـ object نفسه ولو غيرنا في خصائص الـ object داخل الـ Functions نجد أن الخصائص الأصلية للـ object قد تغيرت .



وللملاحظة : تعتبر الأزرار والجرافيكس والموفي كليب و الصوت والفيديو

object لذا فأي تغير يحدث عليها داخل الـ Functions يحدث على الـ object الأساسي وليس على القيمة .

مفهوم الـ **scope** :

هو مدي أو مجال رؤية الـ Compiler لمتغير أو Functions معينة في مدى معين .

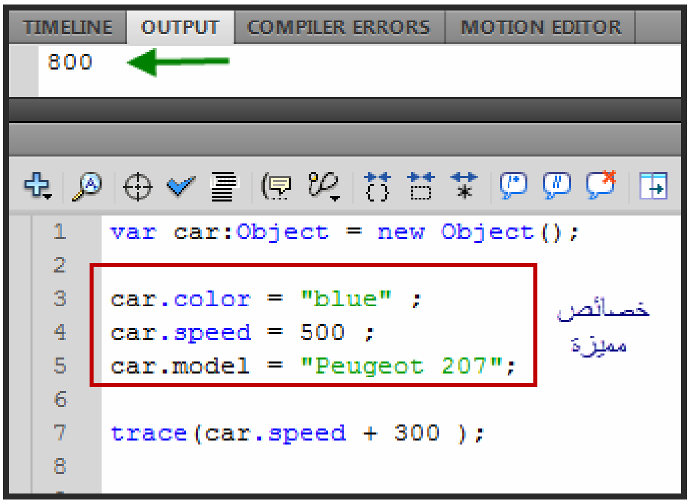
- أي أن المتغير إذا تم تعريفة داخل الـ Functions لا يستخدم إلا داخل الـ Functions الخاصة به فقط ، هذا يطلق علية مصطلح **scope** بينما المتغير المعرف خارج الـ Functions أستطيع إستخدامه وإجراء عمليات علية داخل أي Functions أخرى .
- و لا يعد تعريف متغير جديد - داخل أحد الـ Functions - بنفس أسم المتغير المعرف داخل Functions أخرى تكرار لنفس المتغير السابق حيث أنة كلا خاص بالـ Functions المعرف بداخلها .
- أخذنا قبل ذلك أنة داخل الـ Functions أستطيع استخدام Functions أخرى ولكن للعلم أيضاً انك يمكنك تعريف Functions داخل Functions أخرى ولكن بشرط استخدامها فقط داخل الـ Functions المعرفة داخلها ولا تستخدم من خارج الـ Functions وهذا يسمى **scope** .

الفصل الثامن
الكائنات والفئات البرمجية
Classes & Objects

الكائنات: Objects

كلمة objects تطلق علي أي شئ كائن جماد كان أو حيوان أو إنسان ، ويمتاز كل كائن بمجموعة من الصفات تميزه عن غيره من الكائنات وبالنسبة للغة البرمجة تعتبر هذه الصفات البيانات التي تميز كل كأن عن الأخر كالطول والوزن والحجم والسن والى آخر ذلك من بيانات نقوم بتخزينها للتعامل معها وقت الحاجة وتشبيه الكائنات بأشياء مشابهة للواقع هو مفهوم مشهور في البرمجة يسمى Object Oriented Programming أي البرمجة الكائنة التوجه بمعنى أنها تحاول أن تمثل كل شئ موجود في البرنامج بنفس صفاته الموجودة في الواقع حتى يسهل التفكير في البرنامج

مثال : فرضا أنك تقوم بتصميم لعبة سباق سيارات ، بالطبع كل سيارة تمتاز بمجموع من الصفات تميزها عن غيرها من السيارات بمجموعة صفات أو بلغة البرمجة بيانات مثل لون السيارة وسرعتها ونوعها ، نحتاج لتخزين تلك الصفات داخل البرنامج حتى نستطيع تمييز السيارات عن بعضها ، إذا ما الذي سبق واستخدمناه لتخزين البيانات ؟ إنها المتغيرات ، تعالى لنشاهد ذلك في الكود



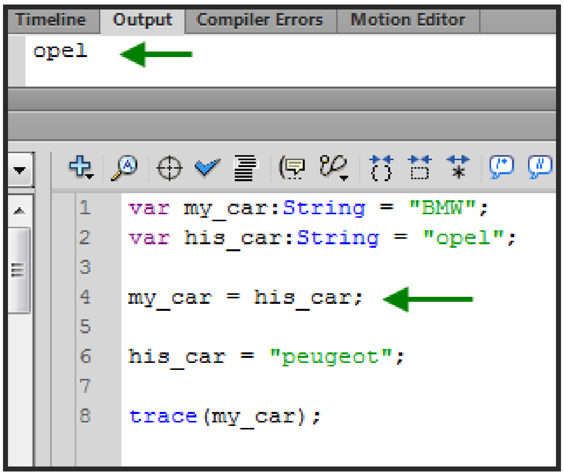
لربط عدد من الصفات بالكائن (Object) الخاص بها كان لا بد من تعريف متغير جديد من النوع Object ، وحتى يعرف ال Compiler بوجد كائن يتوجب حجز مساحه له في الذاكرة يخزن فيها صفات هذا ال Object كان لا بد من استخدام أداة new Object .

بعد تعريف ال Object تأتى مرحلة تخزين الصفات بداخلة من خلال كتابة أسم ال Object والذي في مثالنا هذا هو car ثم نقطة . ثم تحديد نوعية الخصائص المميزة لـ Object سواء أكانت لون أو سرعة أو نوع ثم تخزين البيانات الخاصة بكل صفة .

يمكننا بعد ذلك استخدام أي من البيانات الخاصة بصفات ال Object وتطبيق العمليات الحسابية والمنطقية عليها .

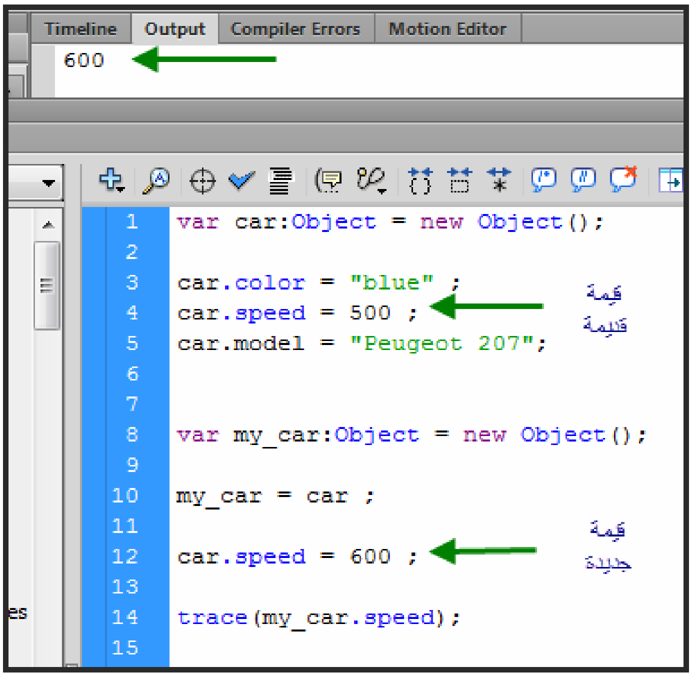
الجميل أنة أصبح في الإمكان عمل عمليات كاملة على ال Object كله مره واحده بكل خصائصه.

ربما أن الظاهر في طريقة التعامل مع ال Object وكأنها نفس طريقة التعامل مع المتغيرات من حيث النظر إلى الكائنات وكأنها نوع من المتغيرات ، إلا أن في الواقع يوجد اختلاف بين المتغيرات والكائنات في طريقة التعامل من حيث



في مثالنا هذا عرفنا متغيرين احدهما my_car والأخر his_car وقمنا بتخزين القيمة "BMW" بداخل المتغير الأول وتخزين القيمة "opel" بداخل المتغير الثاني ، هنا كل متغير يختص بقيمته ، ولكن فرضاً إننا قمنا بتخزين قيمة المتغير his_car داخل المتغير my_car فهذا ليس معناه أن المتغير الثاني نسخة من المتغير الأول يتغير بتغير قيمة الأخر ولكنه يدل على القيمة المخزنة بداخلة فقط ، فإذا قمنا بتغيير قيمه المتغير الأول فإنه لن يحدث شئ للمتغير الثاني .

يختلف ال Object عن المتغيرات في الفكرة السابقة كما يظهر في الشكل التالي .



عند تعريفنا لل Object باسمه car قمنا بإعطائه بعض الخصائص ثم قمنا بتعريف Object جديد باسم my_car فهذا ليس معناه أن my_car هو كائن مستقل عن car بل هو يعتبر بمثابة reference أي مرجع لنفس الشئ بدليل عند تعديل خاصية السرعة للمتغير القديم car وكشفنا عن السرعة باسم المتغير الجديد my_car كانت النتيجة مطابقة لقيمة المتغير الأخر car .

الفئات البرمجية (Classes)

ببساطة يعد الـ class بمثابة التخطيط للـ object .

مثلا :لو أننا نقوم بتصنيع سيارة فإننا نحتاج أولا لرسم هندسي تخطيطي يبين شكل السيارة وعدد أبوابها ولونها وما إلى غير ذلك ، وبتطبيق هذا على درسنا نقول أن السيارة هي الـ object والتخطيط هو الـ classes أي أننا ننشأ classes لننشأ منها object تستخدم في تطبيقاتنا وبمعنى أخر يعد الـ object النسخة الحقيقة من الـ classes والقابلة للتعامل معها والتى تخزن في الـ memory .

من البديهي أن يستخدم التخطيط الواحد لإنشاء أكثر من سيارة ، وكذلك الـ classes تستخدم لإنشاء أكثر من object ، فمثلاً إني في احتياج لعمل سيارة على تخطيط موجود مسبقاً لسيارة أخرى في هذه الحالة أقوم بعمل classes أحدد فية الخواص الموجودة وأنواعها وقيمها الافتراضية التي ممكن تغيرها بعد ذلك .

كون الـ object يستطيع أن يحمل خواص كذلك يستطيع أن يحمل Functions ، ففي مثالنا عن السيارة نجد أن مثلما للسيارة خواص معينة تميزها من لون وسرعة وموديل كذلك لها وظائف معينة تقوم بها من زيادة سرعة وتقليل سرعة وتوقف إلى أخر ذلك من وظائف مرتبطة بالـ object .

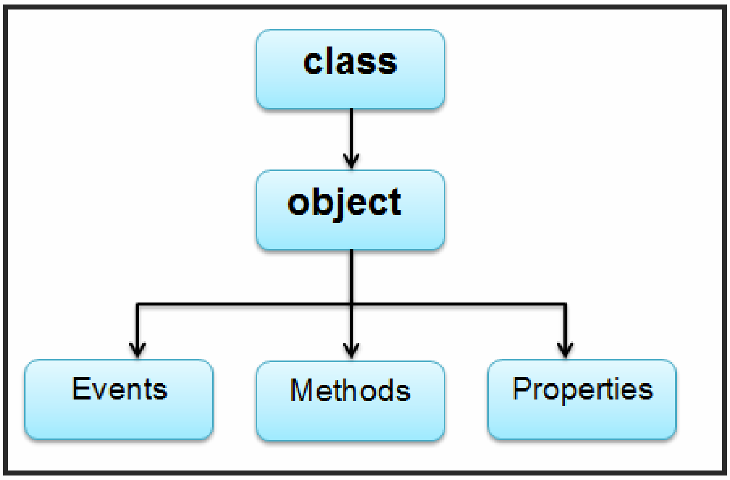
إذا عند تصميم الـ classes أحدد الخواص والوظائف المفترض وجودها في أي object ينشأ من هذا الـ classes .

ولتوضيح فكرة الـ classes والـ object تعالى لنتخيل المثال التالي:

مركز لتعليم اللغات المختلفة (الإنجليزية و الفرنسية و الألمانية)، وفي قسم اللغة الإنجليزية يوجد عدد من المدرسين يشتركون في شئ واحد وهو أنهم يتبعون قسم واحد أو فئة واحدة وهى فئة مدرسين اللغة الإنجليزية ، ويمتاز كل مدرسة بمجموعة من الصفات تميزه عن الأخر كشكله ولونه وأسمه ومواعيد حضوره ، كما أن لكل مدرس مجموعة من الوظائف يقوم بها كالقيام بالتدريس وكيفية تمرين الطلاب على القراءة والكتابة والاستماع فكلها وظائف وآليات كل مدرس يتقنها جيداً ويعرف كيف يقوم بها وكل مدرس أيضا له القدرة على الاستجابة لأي سؤال يطرحه الطلاب علية .

وتطبيقا لمثالنا هذا في تعاملنا مع الأكشن سكريبت نقول أن الفئة التي يتبعها المدرس يماثلها في الكود مفهوم الـ class والمدرس يماثله في الكود مفهوم الـ object ، وكما أن لكل مدرس صفات و وظائف و أحداث يستجيب لها ، كذلك الـ object له الخصائص Properties والآليات Methods والأحداث Events .

وهذه ببساطة هي الفكرة التي قامت عليها البرمجة الكائنية التوجه ...



وسوف نتعلم في درسنا هذا كيفية استخدام الـ classes الجاهزة في برنامج الفلاش لعمل تطبيقات متميزة .

الواقع أننا تعاملنا مع الـ classes من قبل هذا في درس الـ object وفي درس الـ Arrays
وحيث أن الـ Arrays هي نوع من الـ object فإننا نصممه على classes أسمة Arrays

وفي مثالنا التالي قمنا بعمل object من الكلاس Arrays وهو كلاس جاهز في الفلاش
وبالطبع لهذا الـ object خواص مثل خاصية طول الـ Array (trace(animals.length)

```
var animals:Array = new Array("cat","rat","lion");

trace(animals.length);
```

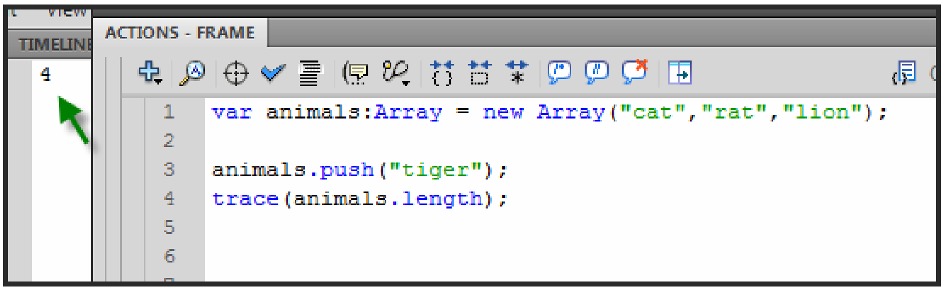
```
var cat:Object = new Object();
```

وحتى عند عمل object كما في الكود السابق قمنا بعمل object أسمه cat من الكلاس المسمى object

- ومثلما للـ object خواص كما سبق وذكرنا فأيضاً لها methods أو وظائف تؤديها ، فتعالى نطبق كلمنا هذا على الـ Arrays ، هل للـ Arrays وظائف ؟ بالطبع للمصفوفات Functions تؤدي وظائف خاصة بالمصفوفات .

للوصول إلى Functions داخل object بنفس طريقة وصولي للـ property

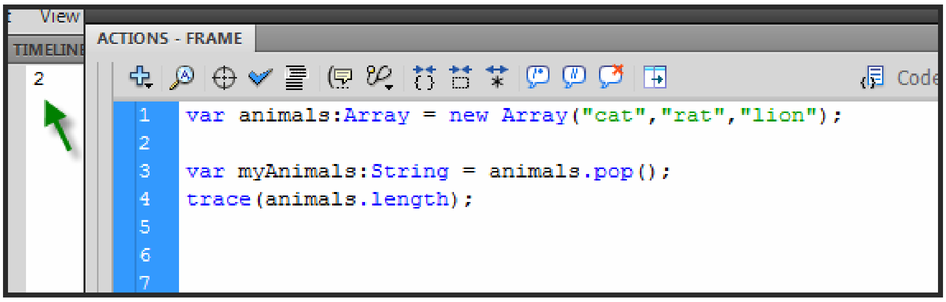
نكتب أسم الـ object ثم نقطة ثم أسم الـ Functions و هنا أخترنا الوظيفة Push لإضافة عنصر زيادة للمصفوفة ونكتب ما بين القوسين أسم العنصر المراد إضافتة .



الوظيفة pop لحذف أخر عنصر من المصفوفة

```
animals.pop();
```

وفي نفس الوقت ممكن استخدام نفس الوظيفة لإرجاع عنصر من خلال تخزيننا للوظيفة pop بداخل متغير كالتالي:



في الواقع أن كل التطبيقات تحتوي على object من كلاس ، وكل object تحتوي على property و methods .

مثلا : الموفي كليب هو عبارة عن object من كلاس اسمه movieclip ، وهذا الـ object لها خصائص تميزه مثل لونه وشكله ومقاسه وأحداثياته ودرجة شفافيته ، كما أن لهذا الـ object وظائف يقوم بها مثل play ,stop وما إلى غير ذلك .

سؤال :

هل كل الـ classes لاستخدامها أحتاج أعمل منها object ؟

الإجابة نعم ولكن لأن لكل قاعدة شواذ لذا فهناك نوع من الـ classes يسمى (statics classes) أستطيع إستخدامه بدون أن اعمل منه object .

statics classes نوعان :

هي classes يمكن استخدامها مباشرة بدون عمل object منها ، وهي تستخدم لأنه قد يكون عندي property أو methods خاصة بشيء معين ولكنها تستخدم في أكثر من مكان في التطبيق ولا أكون في حاجة لعمل object في كل مكان في التطبيق من اجل استخدام الخصائص والوظائف هذه لذا استخدم statics classes حتى أستطيع استخدام الـ property أو methods مباشرة وبسهولة والتي غالباً ما تخص أشياء عامة .

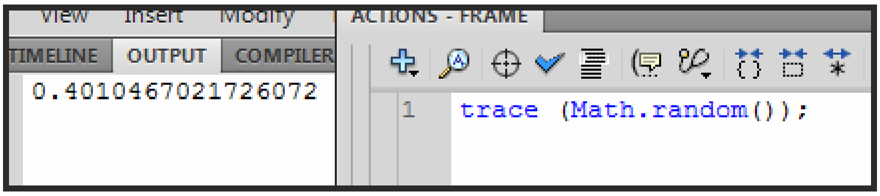
والواقع أن وظائف وخصائص السيارة هي خاصة بشئ معين وهو السيارة ولكننا نستخدم الـ statics classes من اجل الخصائص العامة مثل العمليات الحسابية والهندسية والتى نستخدمها في أغلب تطبيقاتنا مثل الألعاب والمواقع والبرامج وهنا نستخدم math class .

math class وهو من أشهر الكلاس المستخدم في الفلاش بكثرة حيث يحتوي على property و methods الخاصة بالعمليات الحسابية والهندسية .

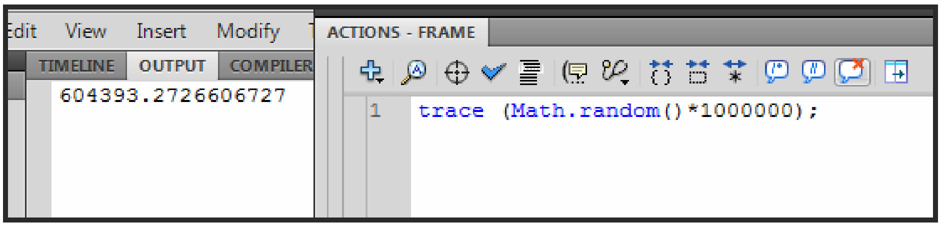
لاستخدام property أو methods داخل الـ math class ، نكتب أسم الـ math class مع ملاحظة أن أول حرف في كلمة math يكون كابيتال ثم نقطة وأسم الشئ المراد أستخدامة .

Math.random();

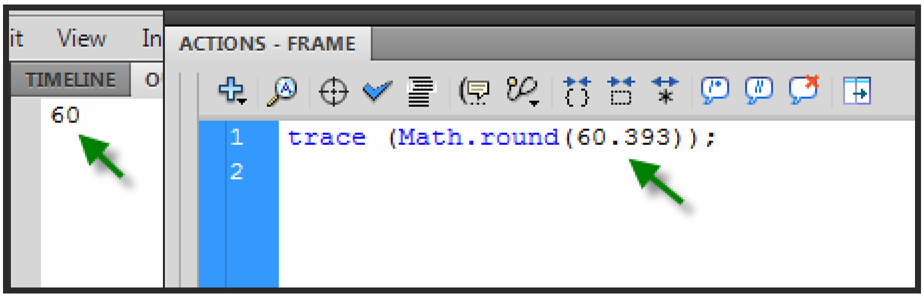
Random هي من أشهر الـ methods المستخدمة مع الـ math class حيث تعطي رقم كسر عشوائي ما بين 0 إلى 1 وكل مره نختبر فيها البرنامج يخرج لي رقم عشوائي مختلف



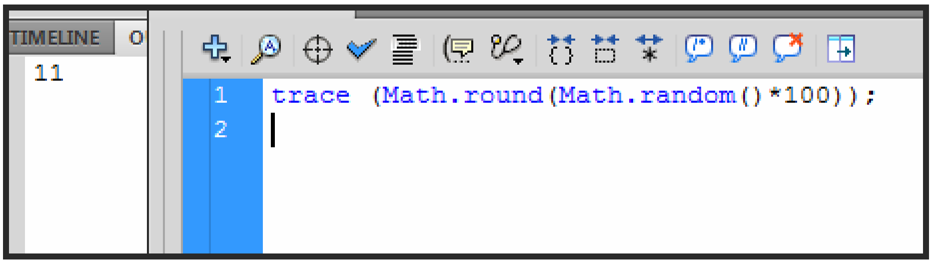
وكما هو ظاهر من أسمها تستخدم هذه الـ methods لعمل شئ عشوائي في البرنامج أو اللعبة وهنا سوف تسألني كيف لوظيفة ما بين رقم 0 إلى 1 أن استخدمها ، في الواقع أنني استطيع استخدام هذه الوظيفة لتوليد رقم عشوائي ما بين أي رقمين ولو فرضنا إني أريد رقم عشوائي ما بين 0 إلى 1مليون هنا نقوم بضرب الوظيفة Math.random * 1000.000



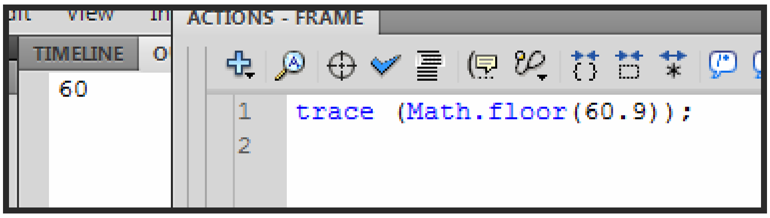
- round هي من الـ methods الشهيرة التي تستخدم مع الـ math class لتقريب الرقم العشري إلى رقم صحيح



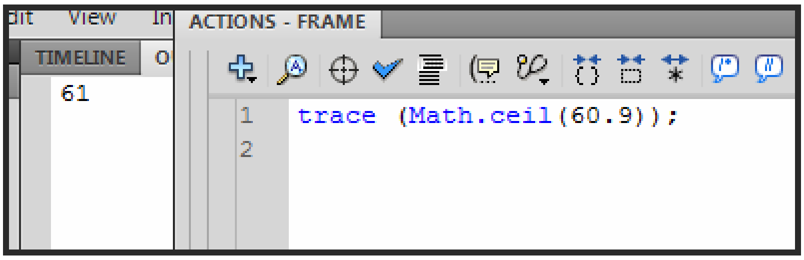
وبالتالي من الممكن استخدام round لتقريب القيمة الكسرية الناتجة من العملية السابقة random لتوليد رقم عشوائي ما بين 1 إلى 100 ولكنه رقم عشوائي صحيح .



- Floor تستخدم مع math class للتقريب إلى اقل رقم صحيح



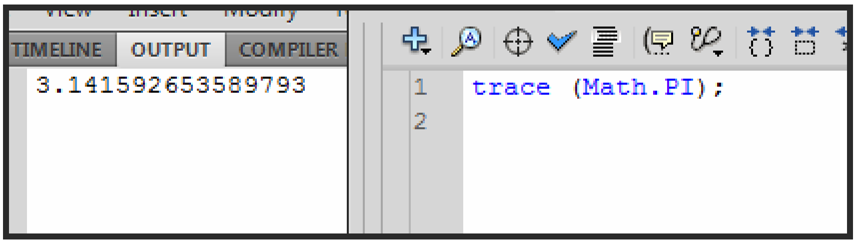
- Ceil تستخدم مع math class للتقريب إلى اكبر رقم صحيح



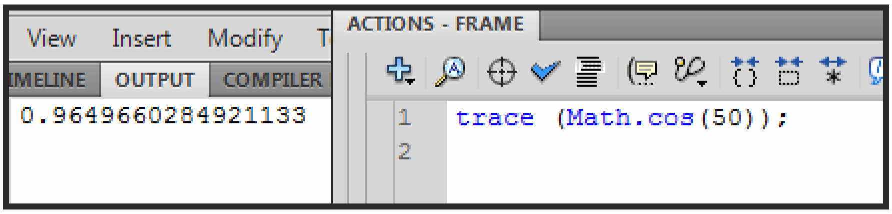
هذا ما كان بالنسبة لـ methods الشهيرة المستخدمة مع math class

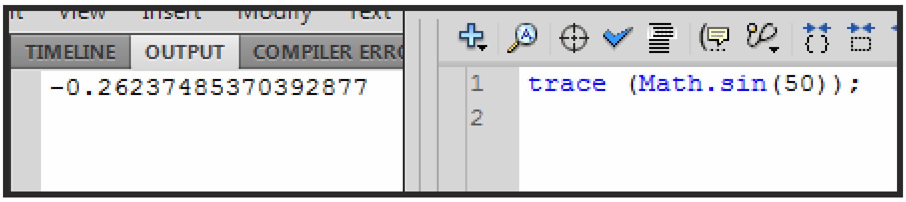
أما بالنسبة لـ property المستخدمة مع math class هي عبارة عن القيم الثابتة الطبيعية مثل :

– PI من الـ property المستخدمة مع math class لحساب محيط الدائرة و هي القيمة الطبيعية المعروفة 22 / 7



- sin ، cos وهي من الدوال الهندسية المعروفة





ملاحظة:

الـ Variable بداخل الـ object تسمى property

الـ functions بداخل الـ object تسمى methods

الفصل التاسع
الكائنات البصرية
Display Objects

Display Objects:

هي المكونات البصرية للتطبيق الذي تقوم ببناءة مثل : photo و graphic و movie clipو text و button وshape و ideo وهي تتشارك في خواص معينة موجودة في كلاً منهم مثل الأحداثيات والطول والعرض وبالطبع هناك خصائص قد تميز Objects عن الأخر مثلما الحال مع الفيديو أو النص الذي يمتاز بخصائص تميزه عن غيره من Display Objects .

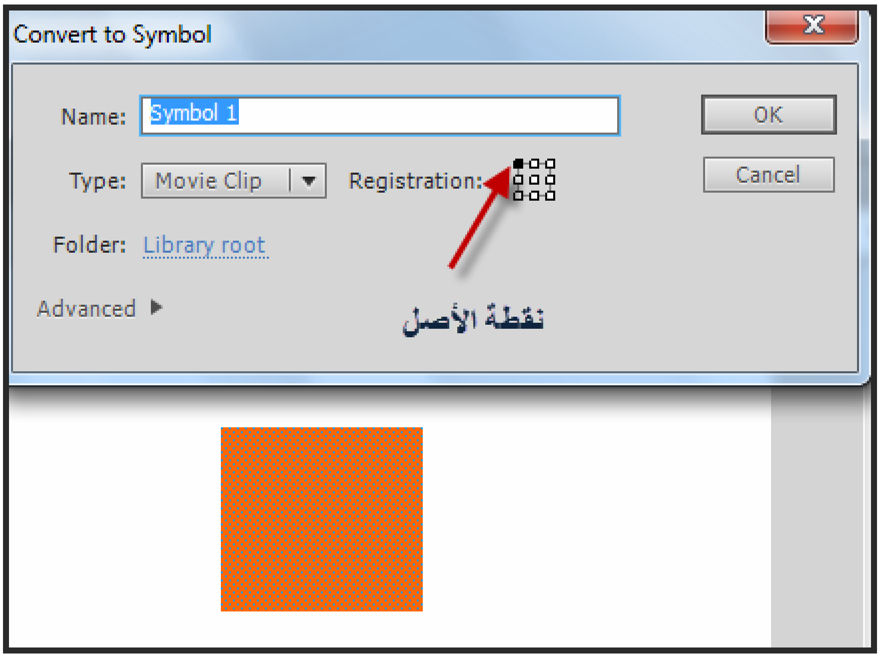
كيفية التحكم في الـ Display Objects من خلال الأكشن سكريبت :

نستطيع التحكم والتعامل مع الـ Display Objects من خلال 3 أشياء وهم(الخصائص Properties والآليات Methods والأحداث Events)، وسوف نتحدث في هذا الفصل عن الخصائص والآليات وأجلنا الحديث عن الأحداث للفصول القادمة .

أولا - الخصائص (Properties)

تعالى لنشاهد الأمر بشكل عملي من خلال عمل Display Objects من النوع movie clip وأخر من النوع button ونطبق عليه الأكشن .

- نقوم برسم مربع مثلاً ثم من مكان وقوفي على الشكل كليك يمين ونختار convert to symbol ونختار movie clip ونختار من أمام registration نقطة الأصل للموفي كليب والتى من خلالها نحدد الحد السيني والصادي للموفي كليب .



- ثم نرسم مستطيل بلون مختلف لتمييزه و كليك يمين ونختار convert to symbol ونختار button .
- للتعامل في الـ movie clip و button السابق إنشاءهم في الكود يجب أن تأخذ أسم ، نقف على الـ movie clip ومن لوحة properties وأمام خانة Instance Name نكتب الاسم المناسب ولیکن box_mc ويفضل تزييل الاسم إذا كان موفي كليب بحرفين mc حتى يسهل تمييز الكائن أثناء قراءتي للكود .

وهناك أيضا مقاطع أخرى مثل:

- **_btn** : لرمز من نوع زر.
- **_txt**: للحقول النصية المتغيرة .

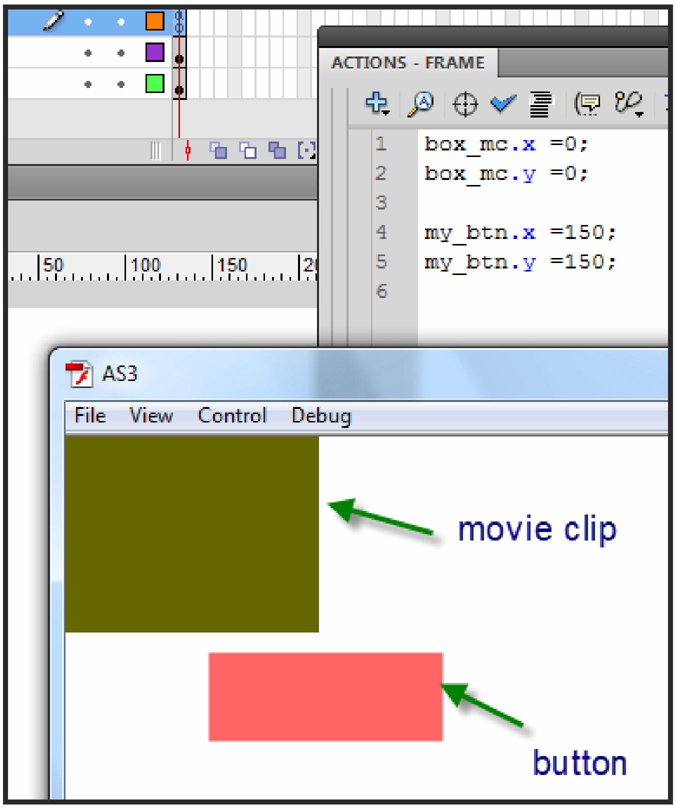
- **_sound** : لملفات الصوت.

وما سبق للموفي كليب نفعله مع الزر فنسميه مثلا my_btn وتم تزيل الاسم بـ btn لتمييزه أنه عبارة عن button .

للتعامل مع خصائص الـ Objects ، نكتب أسم الـ Objects ثم نقطة ثم أسم الخاصية المراد الوصول إليها.

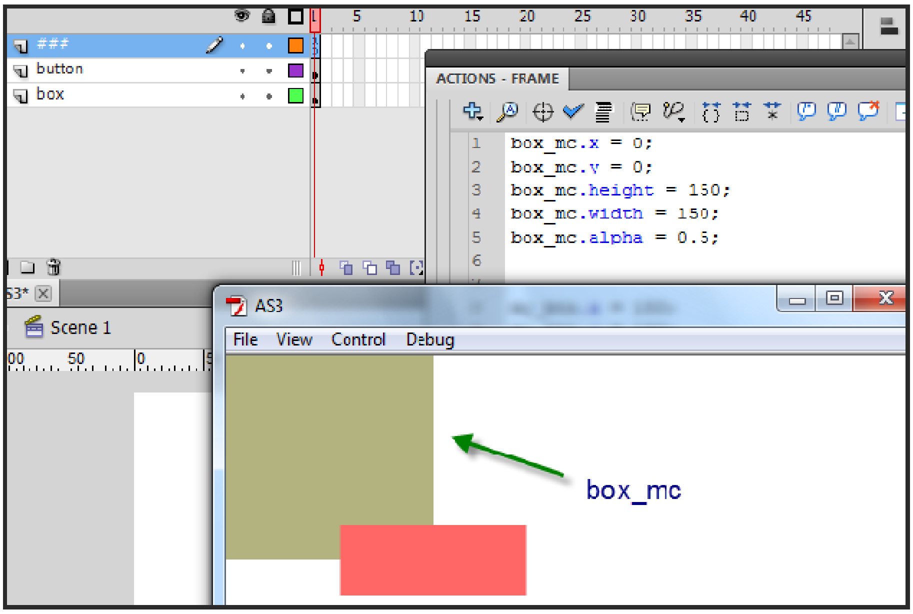
وفي مثالنا هذا نريد أن نعدل في مكان الـ Objects من حيث الحد السيني الأفقي X والحد الصادي الرأسي Y .

كل ما علينا هو تحديد القيمة المراد نقل مكان الـ Objects إليها مع مراعاة أن القيمة تكون بالنسبة لنقطة الأصل الخاصة بالـ Objects .

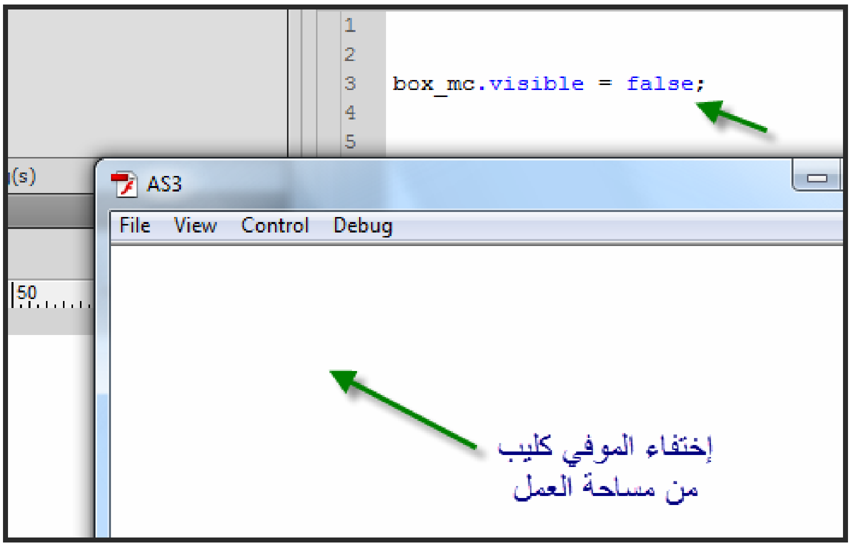


يمكننا أيضا تعديل خواص الـ Objects من حيث الطول(height) والعرض(width).

ويمكن التحكم في مدي شفافية الشكل (alpha) وتتراوح نسبة شفافية الشكل ما بين 1 : 0

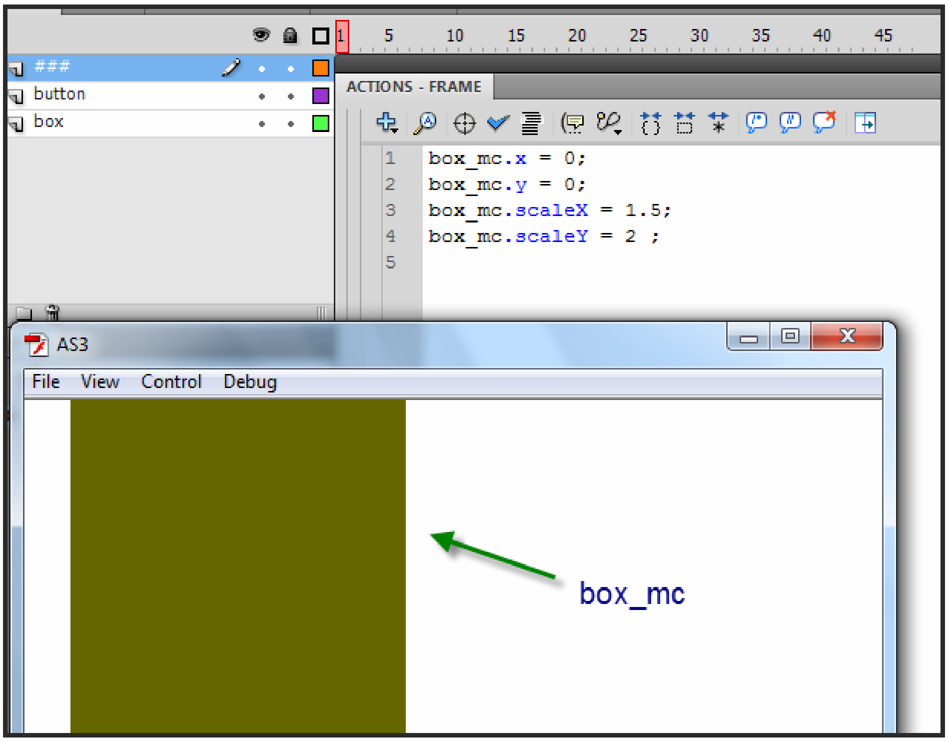


- وهناك خاصية visible والتي تحدد مدى ظهور أو اختفاء الشكل من مساحة العمل وهي خاصية منطقية Boolean أي أنها تقبل أما أحدى قيمتين إما true لظهور الشكل أو false لإخفاء الشكل .



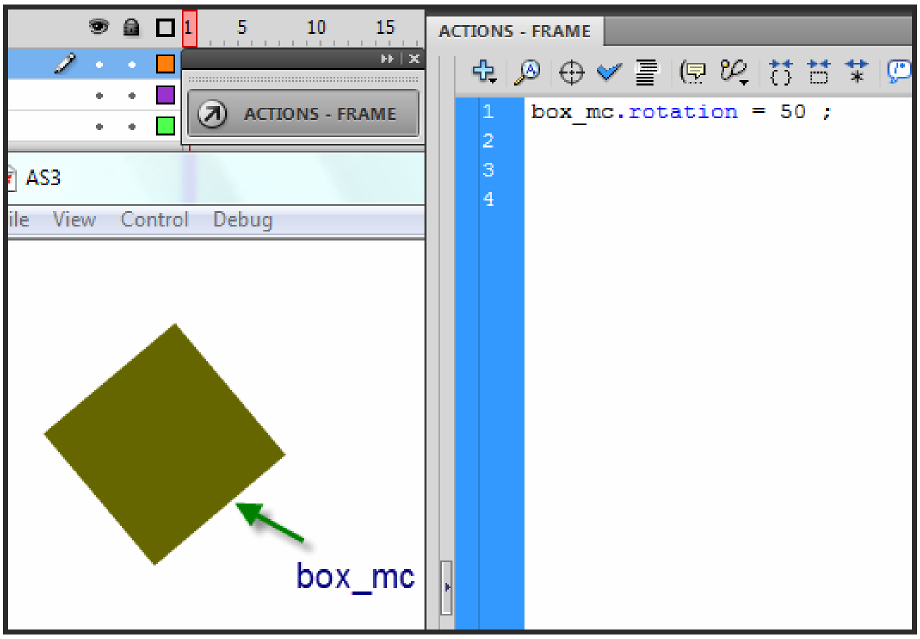
وإخفاء الشكل معناه إخفاءه من مساحة العمل ولكن في الـ memory البرنامج هو ما زال موجود ويمكن إسترجاعه وظهوره في منطقة أخرى من الكود من خلال إعطاء القيمة true .

- ومن الخصائص الشهيرة للـ Objects تعديل العرض(scaleX) أو الطول(scaleY) للشكل بالنسبة للعرض الأصلي الموجود في المكتبة ، ولاحظ أن حرفي X و Y كابيتال وبفرض أننا حددنا القيمة المعدلة للعرض الأصلي = 1.5 فهذا معناه زيادة على العرض الأصلي للموفي كليب بنسبة 1.5 .

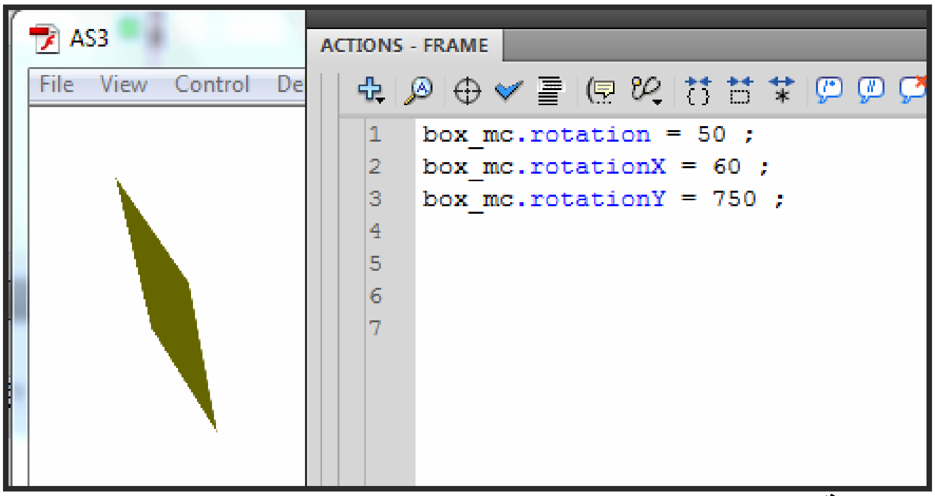


- Rotation :

هي من الخواص المستخدمة لتدوير الـ Objects حسب الرقم المحدد ، والتدوير في الكود يكون حول نقطة الأصل للموفي كليب أو الزرار أثناء تصميمه ، ويمكن تعديل نقطة الأصل عن طريق السحب بالماوس ولكن هذا التعديل يكون للرسم داخل البرنامج ولا يعترف الكود إلا بنقطة الأصل الأساسية والمحددة أثناء تصميم الرمز .

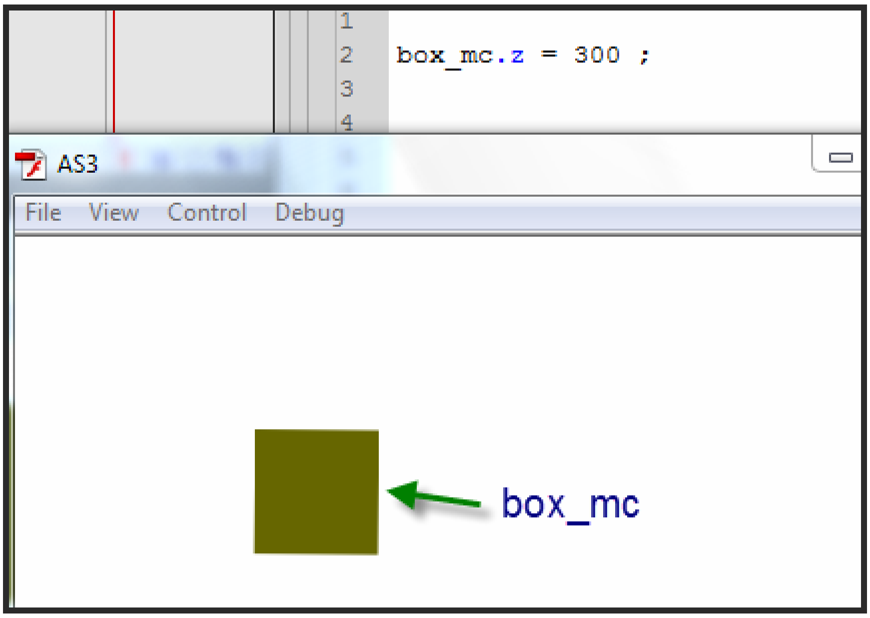


ومن الإصدار العاشر لفلاش (cs4) أصبح في الإمكان عمل تدوير للشكل على محاور ثلاثية الأبعاد مثل التدوير حول المحاور X , Y, Z



ملحوظة:

rotationZ هي نفسها rotation العادية لأنها تدوير للشكل حول نفسه ، ولكن الشغل على المحور Z بدون تدوير يتيح لي عمل عمق وبعد للشكل وكما في مثالنا التالي لو كانت القيمة بالموجب يتم بعد الشكل 300 px ، ولو كانت القيمة بالسالب يتم تقريب الشكل حسب البيكسل المحددة .



هكذا تحدثنا عن بعض الـ property الشهيرة للـ object ، وجاء الوقت للتحدث عن بعض methods المستخدمة مع الـ object .

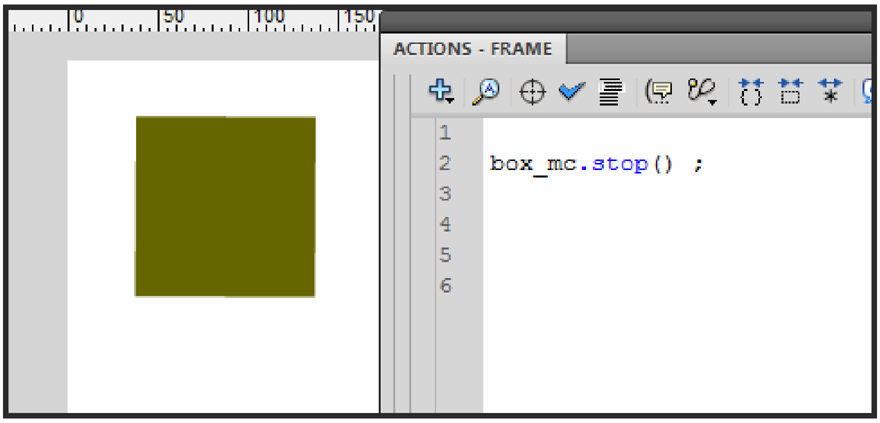
ثانيا - الآليات(Methods)

ومن أشهر الـ methods المستخدمة مع الموفي كليب هي : play , stop , gotoAndPlay , gotoAndStop .

- أستطيع تنفيذ وظيفة معينة مثلما كنت أفعل مع الـ property نكتب أسم الـ Objects ثم نقطة ثم أسم الـ methods المراد تنفيذها .
- Stop :

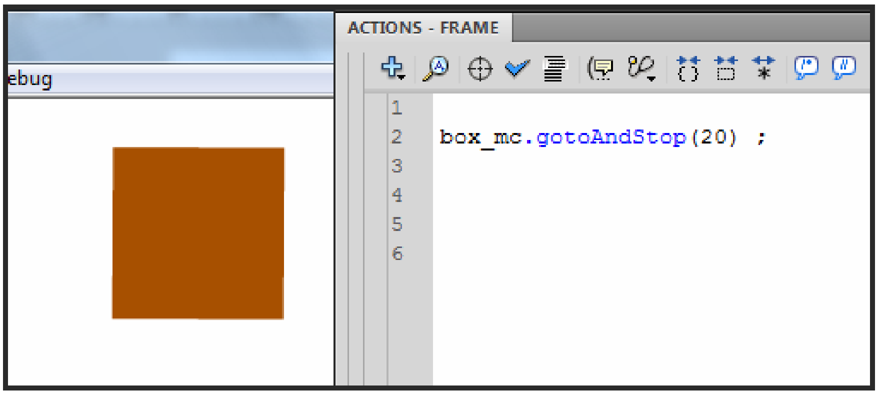
وهي الوظيفة المسئولة عن توقف الموفي كليب الذي يحتوي بداخلة على حركة ، يمكنك تجربة ذلك بجعل المربع الموفي كليب يتحرك ما بين عدد من الـ frames

ويمكن أيضا تغيير لونه عند الكي فريم الأخر ، ولكن بمجرد تنفيذ الوظيفة Stop عليه لا نشاهد أي حركة .



- gotoAndStop :

تستخدم هذه الـ method لتنفيذ توقف عند فريم معين نحدده كما في الكود التالي

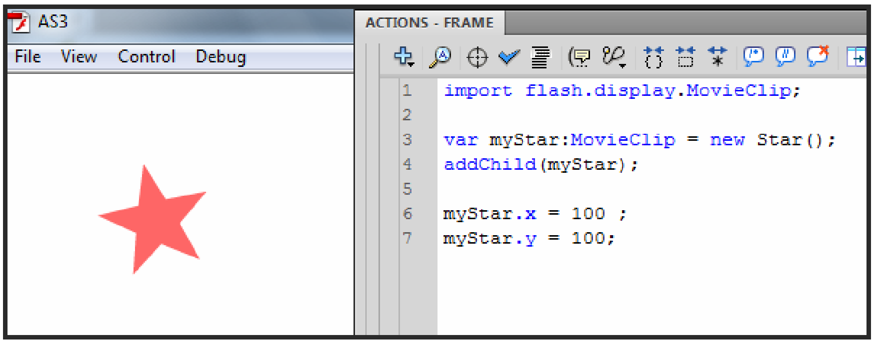


والدليل هنا هو تغيير اللون عند الفريم 20 .

- ما سبق كان تحكم في الـ Objects ولكن على الـ Stage أو بمعنى أخر تحكم في Objects موجودة بالفعل في مساحة العمل ولكننا في أحياناً كثيرةً قد نحتاج لإنشاء واستخدام Objects غير موجود في مساحة العمل وبشكل ديناميك ، كأن نصمم لعبة نستخدم فيها رصاص كثير، هنا لا يشترط وجود الرصاص في مساحة العمل لذا نضطر لإنشائه كنسخ من الـ Objects الأصلي الموجود في المكتبة ، تعالى لنشاهد الأمر بشكل عملي .
- قم بإنشاء شكل لنجمة وحولها لموفي كليب ومن نفس نافذة convert to symbol نضغط على Advanced ليظهر لي مربع حواري أختار تفعيل Export For ActionScript ، ثم أعطي للموفي كليب أسم class معين وكأني هنا أقول للفلاش أعتبر الموفي كليب هذا من ضمن الكلاسات الجاهزة في الفلاش ، واستطيع من class هذا عمل Objects كما هو الحال في حالة عمل Objects من كلاس جاهز في الفلاش مثل الـ Array ثم ok و في

حالة ظهور رسالة تحذيرية بأنني أم اكتب اسم class الخاص بالموفي كليب الخاص بك وتجاهل الرسالة لأنني سوف أستخدم الكلاس الجاهز في الفلاش .

- نقوم بإلغاء موفي كليب النجمة من على مساحة العمل لاختبار كيفية إنشاء Object غير موجود في مساحة العمل .



- نتجه للكود لعمل نسخه من الموفي كليب Star الموجود في المكتبة لوضعه على الـ stage ، وكأني أعمل object من كلاس عادي .
- نقوم في الكود بتعريف object وتسميته بأي أسم وليكن myStar ثم تحديد نوعه و هو هنا من النوع MovieClip ونخزن فيه الموفي كليب باسمة Star .
- عند هذه المرحلة قمت بتعريف object جديد في الـ memory ولكنه لن يظهر على الـ stage (مساحة العمل)
- لجلب الموفي كليب Star لمساحة العمل هنا دور الوظيفة addChild في الكود يليها بين القوسين أسم الـ object الذي سبق وأنشأ ناه و هو باسم myStar

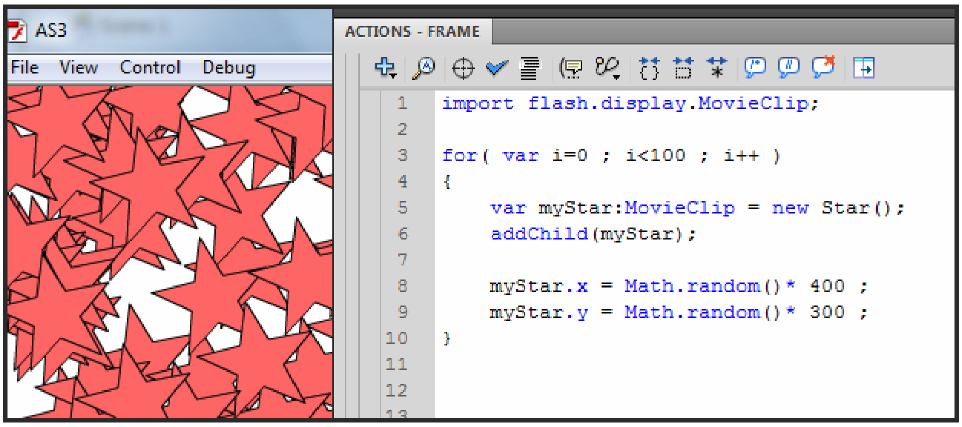
- وباختبار العمل نجده الـ object يظهر في الركن العلوي من الشمال وهذا لأن X , Y غير محددين لذا البرنامج فرضهم بـ 0 ، يمكننا بالفعل تحديد X , Y لـ myStar كما يظهر بالكود .

لعمل أكثر من نسخة من الـ object (myStar) نستخدم for loop لتكرار الشكل أكثر من مرة ولنفرض إلى 100 مرة .

- نكتب for وبين القوسين:

نعرف متغير i كعداد ونضع الشرط المراد اختباره ثم التغيير الذي يحدث على المتغير المستخدم كعداد لكسر تكرار الكود .

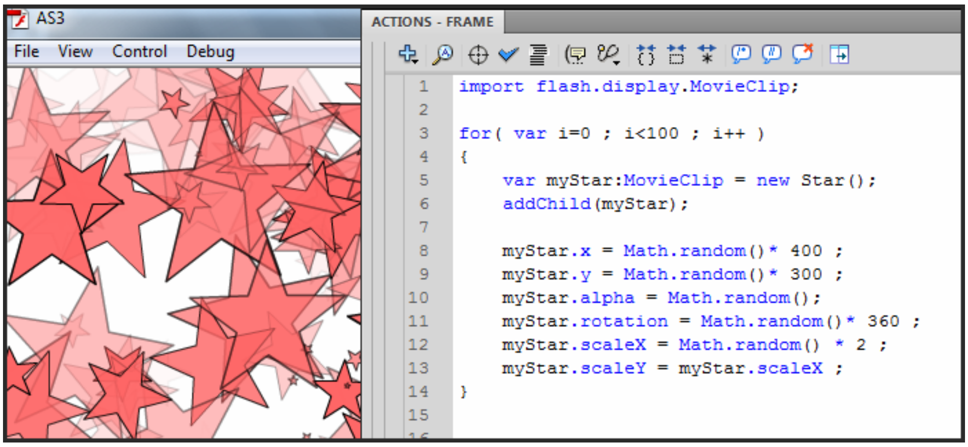
- وبين القوسين المتعرجين نضع جسم الـ Loops أو السطور المراد تكرارها طالما المتغير i اصغر من 100 حتى يتم كسر التكرار .
- عند اختبار العمل نشاهد النجمة واحده ولم يحدث اختلاف عن السابق ، وهذا لأن النجوم المائة نزلت فوق حيث X , Y واحد .
- لذا سوف نعطي لكل نجمة قيمة عشوائية مختلفة عن سابقتها من خلال Math.random (والتى تعطيني رقم عشوائي ما بين 1 : 0)مضروبة في مساحة العمل مرة للـ X بعرض مساحة العمل وأخرى لـ Y بطول مساحة العمل .
- معنى ما سبق هو وضع نجمة لمائة مرة على مساحة العمل مع تغيير X , Y لها عشوائياً.



يمكن العمل على بعض خواص الـ object مثل الشفافية و لجهل قيم الشفافية عشوائية نستخدم أيضا Math.random بدون ضربها في أي قيم حيث قيم الشفافية والعشوائية من 1 : 0 .

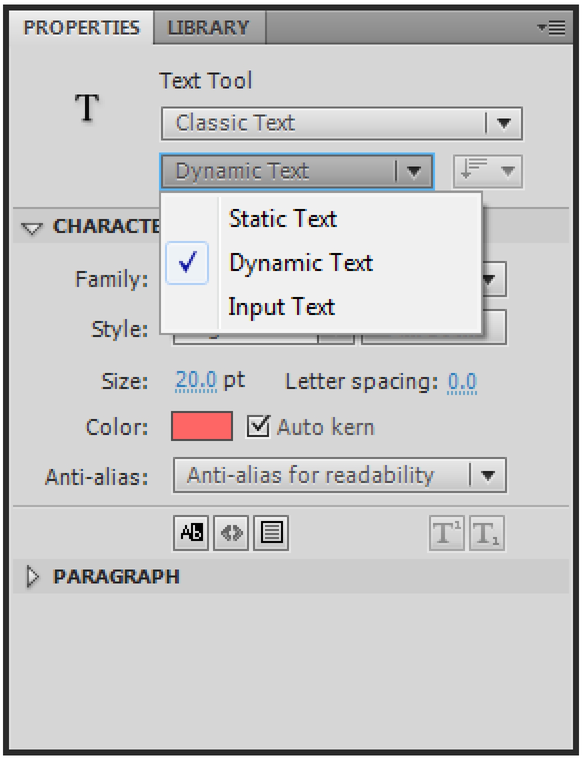
ويمكن العمل علي خاصية تدوير الشكل (Rotation) ولعشوائية التدوير نستخدم Math.random مضروبة في أعلي قيمة لزوايا الشكل و هي 360 درجة .

ومن الخصائص الشهيرة للـ Objects والتي يمكن العمل عليها بشكل عشوائي تعديل العرض(scaleX) أو الطول(scaleY).



Text :

والآن نتناول نوع أخر من الـ Display Objects إلا وهو النص (text) ومن المعروف أن النص في الفلاش 3 أنواع يمكن الاختيار بينهم من لوحة **properties** عند وقوفنا على الأداة **Text** :

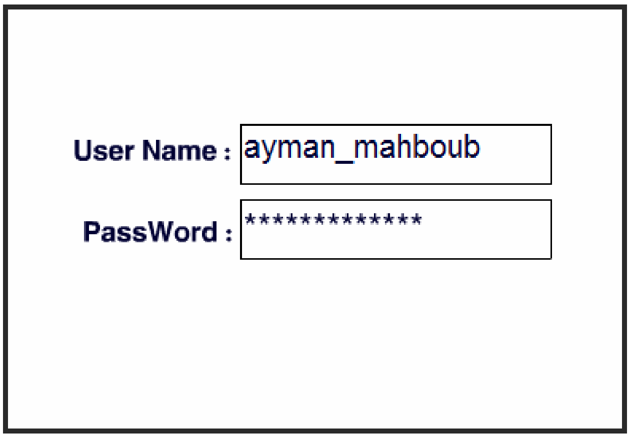


1- Static Text : تستعمل عندما نريد كتابة نصوصا عادية . وهو يختلف عن النوعين التاليين بأنه لا يمكن ربطه بالأكشن سكريبت ولا يمكننا إعطائه اسما Instance Name .

2 – Dynamic Text : تستعمل عندما نريد ربط النص بأوامر أكشن سكريبت كأن نجعل فلاش يظهر نصا موجودا في ملف خارجي وذلك وقت تشغيل الفيلم . وسنأتي على توضيح ذلك فيما بعد أن شاء الله

3 – Input Text : أي نص إدخال وهى منطقة تسمح للمستخدم بالكتابة ، مثلا كأن نطلب منه كتابة بيانات التسجيل ، ويستحسن أن تختار خاصية

Show border around text أي وضع النص داخل مستطيل ، وذلك ليكون هناك مستطيلا فارغا يظهر للمستخدم فيعرف أن عليه كتابة نص بداخله.
كما يمكنك اختيار تنسيق سطر النص بأن يكون في شكل Password ويستعمل هذا التنسيق عندما تطلب من المستخدم إدخال كلمة سر .



عند كتابتنا لنص من النوع Static فإننا لا نستطيع أن نتحكم فية من خلال الأكشن سكريبت ولكن يمكن التحكم في بعض خصائصه مثل مكانه على مساحة العمل وهذا عن طريق تحويله لموفي كليب حتى يسهل التعامل معه .

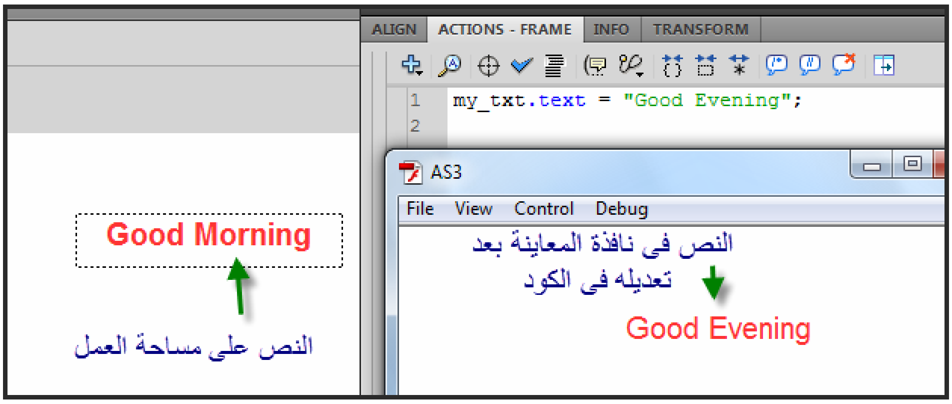
- ولكن لكامل التحكم في القيمة النصية يجب أن يكون الخط من النوع Dynamic .
- ولنفرض أنك تريد تغيير كلمة Good Morning إلى Good Evening ، قم بكتابة الجملة Good Morning بنوع الخط Dynamic ثم سمي الخط في خانة Instance name بأي اسم وليكن my_txt وذلك للتعامل معه من خلال الأكشن ، وكما كان الحال للتعامل مع الـ Objects نتعامل مع النص بنفس الطريقة ، نكتب أسم النص ثم نقطة ثم الخاصية المراد العمل.

- أولا: التعامل مع النص من حيث الـ property :

هناك بعض الخصائص المميزة للتعامل مع النص :

: Text

تستخدم لتعبر عن القيمة النصية الموجودة داخل حقل النص ولو تم تغييرها هنا في الكود سوف تتغير عند عرض التطبيق ، كأن نكتب في الكود Good Evening وهي في مساحة العمل كانت Good Morning

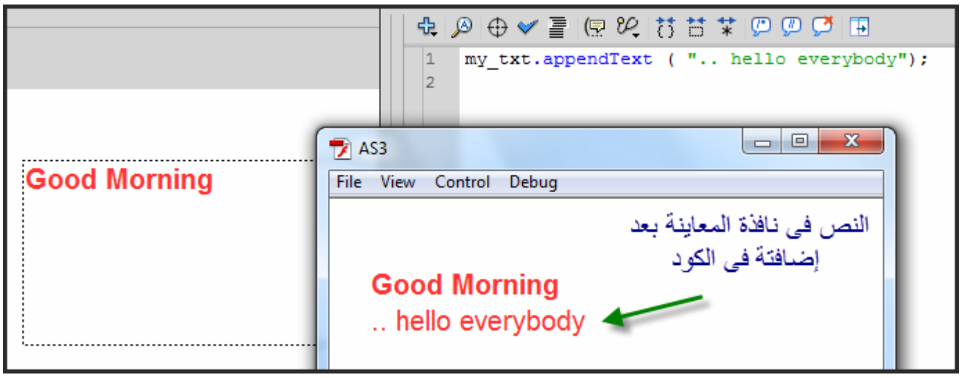


وممكن استخدام هذه الخاصية لعرض عناوين وبيانات لمجموعة صور تحمل من ملف xml خارجي ، أو تستخدم لعرض نص معين عند مرحلة معينة في اللعبة .

وبالنسبة للـ methods المستخدمة مع الـ text

Appendtext ()

تستخدم لإضافة نص زيادة للنص الموجود قبل ذلك ، وبالطبع يجب أعطاء مساحة لحقل النص حتى نستطيع رؤية الكلام المضاف حديثا


my_txt.appendText (".. hello everybody");
Good Morning
AS3
File View Control Debug
النص فى نافذة المعاينة بعد إضافته فى الكود
Good Morning
.. hello everybody

الفصل العاشر
Display List

: Display List

هو النظام المشتمل على القوانين والمبادئ المسئولة عن التحكم وهيكلة الـ Display Objects في ملف الفلاش ، وقد سبق وتحدثنا في الفصل السابق عن Display Objects وعرضنا بعضا من خصائصه وآلياته وسوف نشرح في فصلنا هذا مفهوم Display Objects Container

: Display Objects Container

هو الحاوية للـ Display Objects الأخرى كأن يكون هناك موفي كليب بداخله موفي كليب آخر والفائدة هنا أن الفلاش يضع في الـ Display Objects Container إمكانيات عبارة properties و methods تمكنني التحكم في الـ Display Objects الموجودة في الداخل .
ويعتبر الموفي كليب هو Display Objects Container وتعتبر مساحة العمل (stage) أكبر Display Objects لأنها تحتوي على كل الـ Display Objects الآخرين داخلها .

ما هي الأمكانيات (properties و methods)التي يضعها الفلاش في الـ Display Objects Container للتحكم فية ؟

أولا - الآليات (Methods)

يوضح المثال التالي بعضا من آليات التعامل مع Display Objects Container

قم برسم مربع وحوله لموفي كليب كما تعلمنا ثم ضع في خانة الـ **Instance Name** أسماً له وليكن box_mc ونجعل نقطه الأصل للشكل في المنتصف ، ثم قم برسم دائرة وميزها بلون مختلف عن المربع ثم حولها لموفي كليب وأعطها أسما في خانة الـ **Instance Name** وليكن circle_mc ، ثم تعالى لنتعامل مع كلا الـ box_mc و circle_mc ونطبق درسنا عليهما .

removeChild();

لحذف ابن من داخل الأب أو حذف موفى كليب من داخل موفي كليب آخر

مثال :

لحذف موفي كليب من على الـ stage

removeChild(box_mc);

addChild();

هو إضافة ابن داخل الأب أو إضافة موفي كليب داخل موفي كليب آخر
وبفرض إضافة المربع داخل الموفي كليب الدائرة يكون الكود كالتالي:

circal_mc.addChild(box_mc);

: Depth

للتحكم في ترتيب الـ Display Objects على مساحة العمل وأيضاً للتحكم في ترتيب Display Objects Container ، ويترتب الفلاش الـ Display Objects داخل الـ Display Objects Container عن طريق الـ index حيث كل Display Objects له رقم معين يحدد ترتيبه بالنسبة للـ Display list بمعنى أي Display Objects جديد يظهر فوق الـ Display Objects الأخرين .

والترتيب هنا يشبه ترتيب العناصر في المصفوفات بمعنى أن أخر عنصر وهو أسفل كل العناصر في الترتيب ترتيبه 0 والأعلى1 والأعلى 2 والأعلى 3 وهكذا ولأن كل عنصر له ترتيب معين لذا الفلاش يتيح لك بعض methods لاستغلال هذا الترتيب

ونحتاج كثيرا الترتيب هذا في تطبيقاتنا عندما نريد عنصر فوق الأخر كترتيب معين للصور مثلاً .

مثال :

removeChildAt();

لحذف ابن من داخل الأب أو حذف موفى كليب من داخل موفي كليب آخر ولكن هنا بحسب رقمه وترتيبه داخل الـ Display Objects Container .

مثال :

لحذف موفي كليب بداخل موفي كليب وترتيبه في أسفل كل العناصر أي رقم الـ index الخاص بة 0 يكون الكود كالتالي :

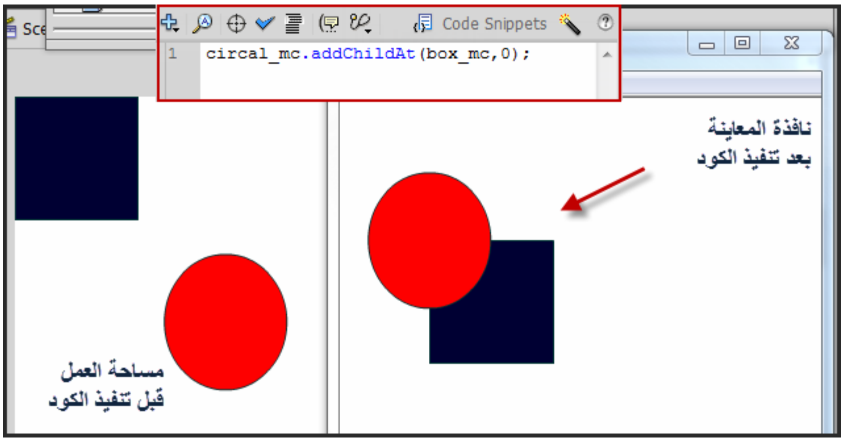
```
circal_mc.removeChildAt(0);
```

addChildAt();

هو إضافة ابن داخل الأب أو إضافة موفي كليب داخل موفي كليب آخر وفى ترتيب معين وبفرض إضافة المربع داخل الموفي كليب الدائرة يكون الكود بكتابة أسم موفى كليب الأب (circal_mc) ثم نقطة ثم **addChildAt()** **والتى تأخذ 2 parameter** الأول هو أسم الموفى كليب الذي أرغب أضافتة داخل الدائرة والثانى رقم الـ index الذي أريد ترتيب الموفى كليب الأبن فية

```
circal_mc.addChildAt(box_mc,0);
```

ويلاحظ بعد عمل معاينة للملف اختلاف مكان box_mc من حيث X و Y وذلك لأن خواص الأبناء نسبية بالنسبة للآباء ، بمعنى أنه عندما كان box_mc على مساحة العمل الرئيسية (Stage) يأخذ 0 لكلا من X و Y كان مكانه في الركن العلوي من الشمال ، وبعد تنفيذ الكود السابق يظهر box_mc تحت الدائرة ملاصق زاوية الدائرة السفلية من اليمين حيث أن قيمتي كلا من X و Y أصبحت بالنسبة لنقطة أصل الدائرة التي هي أب للمربع الابن .



ثانيا - الخصائص (Properties)

سوف نعمل في مثلنا هذا على خاصيتين visible و alpha وبالقياس يمكنك استخدام أيا من الـ Properties التي تناولناها في الفصل السابق .

كل ما ينطبق على الكود مع الـ Time Line الرئيسي ينطبق على الكود مع الـ Time Line داخل الموفي كليب ، ولو دخلنا بداخل أي موفي كليب داخل الموفي كليب الرئيسي نستطيع أيضا التعامل بالكود مع الـ Time Line الخاص بهذا الموفي كليب بنفس الطرق السابقة .

الواقع أنه من الأفضل التفكير في الـ Time Line الرئيسي على أنه عبارة عن موفي كليب كبير يحتوي على كل الـ Display Objects الموجودة على الـ stage .

ويشار للـ Time Line الرئيسي في الكود بكلمة root وتعني جذر حيث يحتوي على كل الـ Display Objects التي يتكون منها ملف الفلاش .

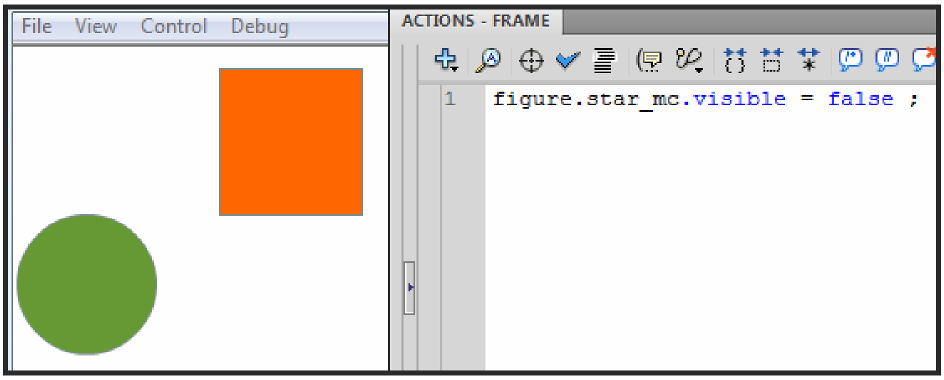
بما انه يمكنني كتابة الكود بداخل أي موفي كليب ، فإنه قد يتطلب الأمر استخدام كود معين من داخل موفي كليب الوصول لموفي كليب أخر

مثال على ما سبق:

قم بعمل موفي كليب باسم مثلاً figure يحتوي بداخلة على 3 أشكال هندسية (مربع ودائرة ونجمة) وكل شكل هو عبارة عن موفي كليب مستقل داخل الموفي كليب الرئيسي figure ثم قم بإعطاء كل شكل أسم في خانة الـ Name Instance حتى تستطيع التعامل معه في الكود (rectangle_mc , star_mc, circle_mc) .

فرضاً أننا نريد التحكم من على الـ stage في أحد الأشكال الهندسية وليكن أخفاء الدائرة circle الموجودة داخل موفي كليب figure ، وكما سبق وذكرنا كل شكل هو موفي كليب منفصل .

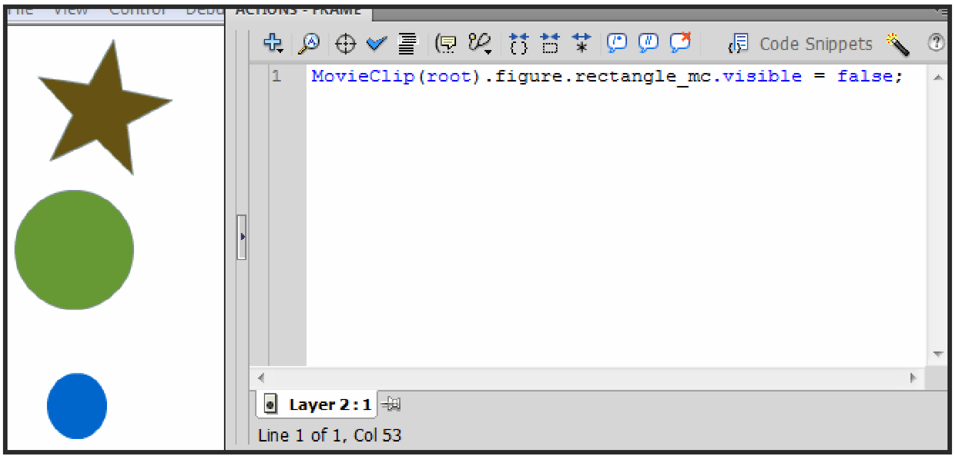
ولأن الدائرة موفي كليب داخل موفي كليب آخر نبدأ في الكود بكتابة أسم الموفي كليب الرئيسي figure ثم نقطة ثم أسم الموفي كليب الداخلي الذي يعد بمثابة ابن للموفي كليب الأب child السابق والمراد إخفاءه ثم نقطة ثم نحدد نوع الـ property المراد التعامل معها بالنسبة للموفي كليب وهي هنا visible وحيث أن هذه الخاصية منطقية تحتمل أما true لظهور الشكل أو false لإخفاء الشكل ، ويكون الكود كالتالي :



ونلاحظ اختفاء شكل النجمة من بين الأشكال المرسمة على الـ stage

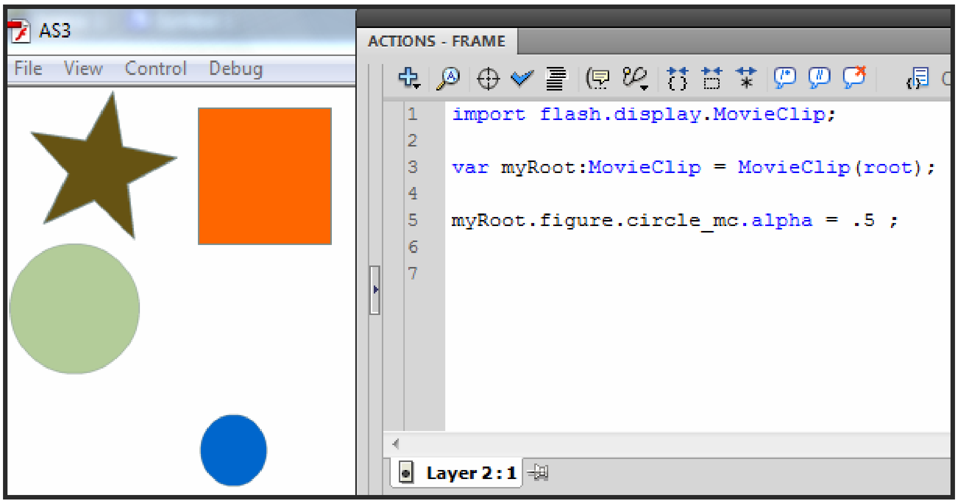
- وفرضاً إننا نريد تنفيذ الكود السابق لإخفاء شكل المربع و لكن من داخل موفي كليب أخر .
- مثال : نرسم دائرة منفصلة عن الأشكال السابقة ونحولها لموفي كليب ونعطيها أسم خانة الـ Instance Name وليكن cc_mc ثم بالنقر عليها مرتين ندخل داخل الموفي كليب ومن على time line الخاص بموفي كليب الدائرة نبدأ بكتابة الكود الخاص بإخفاء موفي كليب المربع .

وحيث انة يشار time line الرئيسي في الكود بالمصطلح root ، وحيث أننا نتعامل مع time line الرئيسي على أنه موفي كليب كبير يحتوي على كل الـ Display Objects الموجودة على الـ stage وحيث يجب لاستخدام الـ root على أنه أب لكل الـ Display Objects الموجودة على الـ stage أن اخبر الفلاش صراحاً أن يتعامل مع الـ root أنه موفي كليب لذا لزم علينا في الكود كتابة كلمة MovieClip سابقة لكلمة root الموجودة بين قوسين



بالطبع يفضل عند كتابة كود داخل موفي كليب أن يكون الكود مختص بالتحكم بمكونات هذا الموفي كليب .

وإذا كنا سنتعامل كثيراً مع الـ root يمكننا تسهيلاً للعمل تخزينها داخل متغير نستدعيه في أي وقت كالتالي :



الفصل الحادي عشر
الأحداث
Events

:Events

هو شيء معين يحدث لتطبيق الفلاش الذي تقوم به ، كأن يضغط المستخدم على زر معين أو يضغط اللاعب بالماوس في مكان معين لحدوث حدث معين ، أو أن يضغط الشخص على أحد أزرار لوحة المفاتيح ، أو كأن يبدأ التطبيق في التحميل أو ينتهي من التحميل ، أو أن يكون هناك فيديو في تطبيق الفلاش فيبدأ التشغيل أو ينتهي من التشغيل ، كل هذا يسمى أحداث .

برنامج الفلاش يعطيك أمكانية التعامل مع الأحداث ومن ثم تأتي فكرة التفاعل لأنه بدون أحداث لا يكون هناك تفاعل ، حيث يعطيني برنامج الفلاش القدرة على تطبيق كود معين عند حدوث حدث معين ، كأن يحدث عند الضغط علي زر معين أن ينفذ كود معين ، والواقع أن الأحداث الموجودة في الفلاش كثيرة جدا وهذا يعطيك سيطرة كبيرة جداً على التطبيق الذي تقوم ببناءة ، وبسبب هذه الأحداث يعرف برنامج الفلاش بأنه منصة تفاعلية قوية جداً ، حيث يعطيني الفلاش القدرة على التحكم في الأحداث سواء كان الحدث ناتج من المستخدم أو أن يحدث الحدث تلقائي من الفلاش كأحداث التحميل ويسمى التعامل مع الأحداث من خلال الأكشن سكريبت بـ Event Handling ، والجميل أنه برغم العدد الكبير من الأحداث إلا أن الفلاش يتعامل معها بنفس الطريقة أو بنفس الأسلوب كما سنرى .

يحتوي فلاش على أنواع كثيرة من الـ Objects مثل الـ Display Objects وهناك Objects من أنواع أخرى وبرغم أن هناك أنواع كثيرة تندرج تحت نفس المسمى Display Objects إلا أنه بينهم خلافات مثل اختلاف الـ Text عن Movie Clip وبالتالي بعض الـ Event مشتركة بين بعض الـ Objects ومختلفة بين البعض الأخر ، كالأحداث الخاصة بالفيديو

بالتأكيد لن تكون موجودة عند التعامل مع النص لأن هذا حدث مرتبط بنوع الـ Objects نفسه .

Mouse Events

وهي الأحداث التي تحدث من خلال الماوس **مثال:**

- نقوم برسم مربع مثلاً ثم من وقوفي على الشكل كليك يمين ونختار Convert to Symbol ونختار Movie Cip ونختار من أمام Registration نقطة الأصل للموفي كليب والتى من خلالها نحدد الحد السيني والصادي للموفي كليب .
- للتعامل مع الـ movie clip - المربع السابق إنشاءه - في الكود يجب أن يأخذ أسم ، نقف على الـ Movie Clip ومن لوحة Properties وأمام خانة Instance Name نكتب الاسم المناسب وليكن box_mc ويفضل تزييل الاسم إذا كان موفي كليب بحرفين mc حتى يسهل تمييز الكائن أثناء قراءتي للكود ..
- ولنرى الآن كيف يستجيب الـ Display Objects لأحداث معينة والجميل أن طريقة الاستجابة أو التعامل مع الحدث كلها تتم بنفس الطريقة وبنفس الخطوات كالتالي :
- نكتب أسم الـ Objects الذي أريد الاستجابة للحدث الخاص به ثم نقطة ثم أسم الـ method الخاصة بالحدث وهي addEventListener وهذه الـ Method تأخذ 2 Parameter :

1- نوع الـ Event الذي أريد الاستجابة له .

2- أسم الـ Function الذي أريد تنفيذها عند حدوث الحدث .

بالتطبيق على مثالنا – بالنسبة لنوع الحدث وهو هنا حدث خاص بالماوس حيث أريد الاستجابة لحدث الكليك بالماوس على المربع ، يعتبر هذا الحدث من ضمن مجموعة من الأحداث تسمى Mouse Event وهي الأحداث التي تحدث من خلال الماوس وهي أحداث كثيرة.

لذا سوف نفتح قوس ونكتب نوع الحدث MouseEvent وهو كلاس يحتوي على مجموعة أحداث الماوس ، ثم نقطة ثم نأتي لأسم حدث الماوس وليكن هنا CLICK ثم فاصلة .

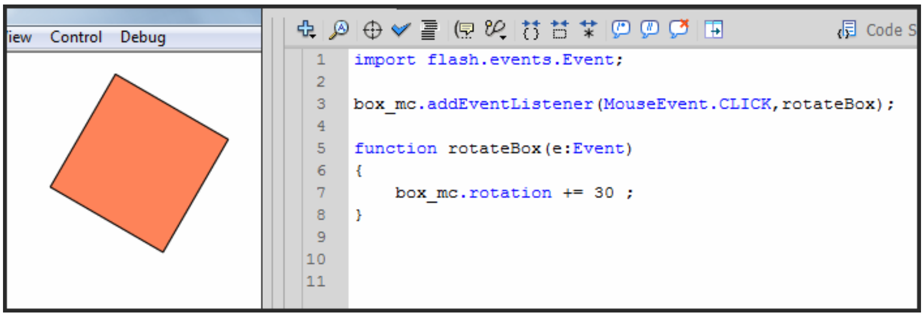
نأتي لأسم الـ Function الذي أريد تنفيذها عند حدوث الحدث ولتكن عمل تدوير للمربع فنكتب rotateBox ثم نقفل القوس .

- لاحظ عدم وضع الوظيفة rotateBox بين قوسين لأننا لا ننفذ هنا هذه الـ Function ولكننا فقط نرسل أسمها لتحدث عندما يحدث الحدث السابق الكلام عنه لذا يستلزم الأمر إنشاء Function باسم rotateBox .
- نكتب Function ثم مسافة ثم أسم الـ Function المطلوبة وهي هنا rotateBox .
- الـ Function التي تستجيب لأحداث تسمى event handler لأنها تتعامل مع الحدث وهي تستقبل parameter يرسله الفلاش له من نوع event بمعنى أنه هناك كلاس أسمة event يرسل Objects موجود بعض المعلومات المهمة عن الحدث قد أستفيد منها

وقد لا أستفيد منها في التطبيق الذي أصممه والبرنامج لا بد أن يرسله ولا بد علي أن أستقبله بغض النظر أن كنت سأستخدمه أم لا ومن المتعارف علية أنه يسمى e ليرمز إلى event ، نضع بين القوسين الحرف e ثم نقطتين لنحدد نوع الـ parameter وهو هنا event

.

- وتعتبر e Objects من النوع event .
- بين القوسين المتعرجين نكتب السطور التي أريد تنفيذها في حالة حدوث الحدث ولتكن تدوير الشكل بزاوية معينة ، نكتب أسم الـ Objects ثم نقطة ثم نوع الحدث وهو زيادة تدوير الشكل rotation بزاوية 30 درجة كل ضغطة بالماوس .



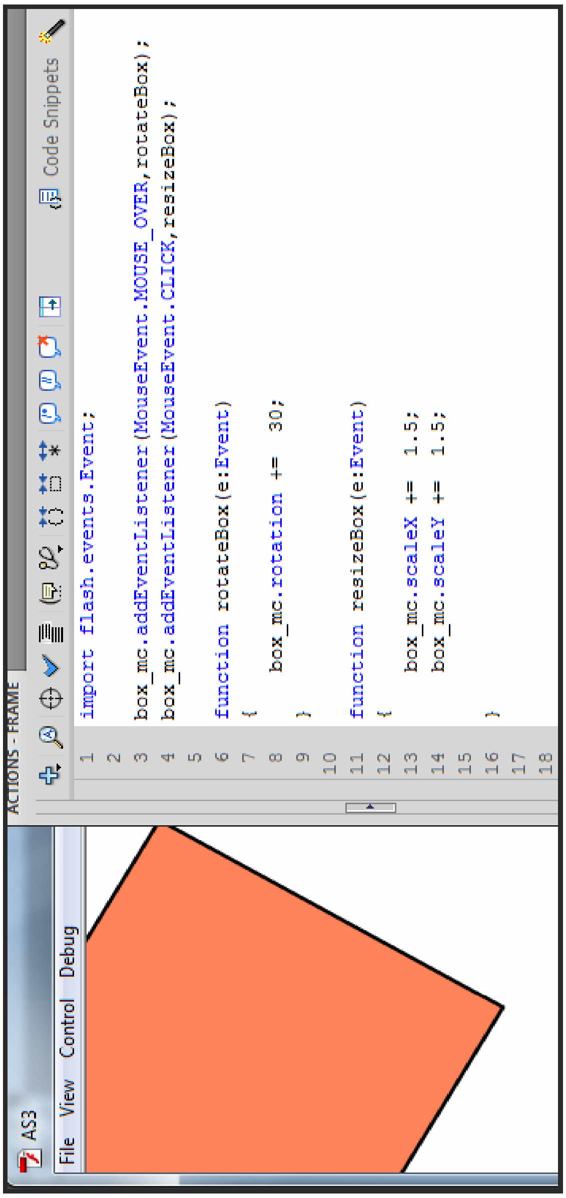
في مثالنا السابق كان حدث الماوس هو CLICK ، أي بمجرد عمل كليك بمؤشر الماوس على الـ Objects ، وبالطبع هناك أحداث أخرى للماوس تعالى لنتناولها أغلب الأحداث المستخدمة :

- **MOUSE_DOWN** : معناها أنه بالضغط بالماوس على الـ Objects ينفذ الحدث قبل رفع أصبعك عن الماوس.

- **MOUSE_UP** : الضغط بالماوس ثم بعد رفع الأصبع عن الماوس ينفذ الحدث .
- **MOUSE_MOVE**: ينفذ الحدث بمجرد تحرك الماوس فوق الـ Objects
- **MOUSE_OUT** : ينفذ الحدث عند الضغط بمؤشر الماوس على الـ Objects ثم الخروج خارج حدود الـ Objects.
- **MOUSE_OVER** : عند الدخول بمؤشر الماوس فوق الـ Objects ينفذ الحدث .

من الواضح أن الأحداث هي سبيلي لعمل تفاعل مع المستخدم .

بالطبع أستطيع الاستجابة لأكثر من حدث لنفس الـ Objects ، كأن أنفذ بجانب عمل تدوير للشكل عمل تعديل في حجمه مثلاً زيادة عن حجمه مرة ونصف ، ومعنى زيادة حجمه عن الحجم الأصلي مرة ونصف بمجرد كليك على الشكل أن هذا الحدث سيحدث مرةً واحدة وإذا كان الغرض هو زيادة الحجم كل كليك بالماوس هنا يجب زيادة العلامة + قبل العلامة =



تعالى لنتعرف على فائدة الـ parameter (e) الذي يرسله الفلاش إلى الـ Function وكيفية الاستفادة منها.

- نعمل موفي كليب جديد بشكل نجمة Star ونعطيه أسم للتعامل معه في الكود وليكن Star_mc ونفرض أننا نريد موفي كليب النجمة أن يستجيب للحدث CLICK بنفس طريقة الاستجابة لهذا الحدث على الموفي كليب Box .
- بالطبع الحل السهل هو أن أعادة نفس الكود السابق والخاص بشكل المربع وتطبيقه على شكل النجمة .

ولكن هناك طريقة أخرى بدلاً من تكرار الكود وهيا استغلال الـ parameter (e) الذي يرسله الفلاش إلى الـ Function ، وهنا بدلاً من تصميم حدث لشئ بعينه يمكننا تصميم أحداث لـ Objects مختلفة .

- تعالى لنكتب الكود السابق بشكل جديد وذلك من خلال استبدال أسم الـ Function الذي أريد تنفيذها عند حدوث الحدث باسم عام وليكن rotateObjects ، resizeObjects
- ثم نأتي لمرحلة إنشاء الـ Function نكتب Function ثم مسافة ثم أسم الـ Function المطلوبة وهي هنا rotateObjects .
- وبين القوسين التاليين لأسم الـ Function نكتب الحرف e ثم نقطتين لنحدد نوع الـ parameter وهو هنا event .
- وعلينا أن نحدد بين القوسين المتعرجين أسم الموفي كليب الذي يستجيب للحدث ، كيف يتم هذا ، ببساطة شديدة الـ parameter (e) له property أسمها target يخزن فيها اتوماتيك الـ

Objects الذي حدث له الحدث بمعنى إذا كان الحدث كان لـ box نجد الخاصية target مخزن بداخلها الموفي كليب box.

- نكتب الحرف e ثم نقطة ثم target ثم نقطة ثم نوع الحدث rotations ثم تخزين الرقم المطلوب حدوثه وليكن 30=+ .

```
box_mc.addEventListener(MouseEvent.MOUSE_OVER,rotateObjects)

function rotateObjects(e:Event)
{
    e.target.rotation +=  30;
}
```

و هنا أياً كان الـ Objects يستجيب حسب الحدث الموضوع له .

أحداث الـ Stage

تعلمنا من قبل أن مساحة العمل Stage تعتبر Display Objects Container يحتوي على كل الـ Display Objects المستخدمة في التطبيق ، وبما أن الـ Stage تعتبر Objects بغض النظر عن نوعه ، لذا فهي أيضا تستجيب لمجموعة من الأحداث من ضمنها معظم الأحداث التي تستجيب لها باقي الـ Display Objects العادية سواء graphic أو text أو button أو movie clip أو video مثل حدث CLICK بالماوس سواء على Objects أو على مكان خالي على الـ Stage تستجيب لها الـ Stage بشكل عادي ، تعالى لنشاهد الأمر بشكل عملي على الكود.

- نضع اسم الـ Objects وهو هنا الـ Stage ثم نقطة ثم أسم الـ method الخاصة بالحدث وهي addEventListener وبين القوسين نضع الحدث للماوس وهو هنا مثلاً CLICK ثم اسم الـ Function ولتكن moveObjects .
- تأتي مرحلة أنشاء الـ Function وهي بنفس الوضعية في الأكواد السابقة ونحدد بين القوسين المتعرجين نكتب السطور التي أريد تنفيذها في حالة حدوث الحدث ولتكن تحريك المربع والنجمة على الأحداثي X ، Y بقيمة 5 كل كليك بالماوس

```

stage.addEventListener(MouseEvent.CLICK,moveObjects);

function moveObjects(e:Event)
{
    box_mc.y += 5;
    star_mc.x += 5;
}


```

- مثلما أحتاج إضافة حدث أحتاج حذف حدث معين بعد مرحلة معينة ، وتطبيقاً على مثالنا السابق وهو كل كليك تزود في الـ Y الخاصة بالمربع وبفرض أننا نريد توقف هذا الحدث عند الضغط على الموفي كليب الأخر وهو النجمة ، تعالى لنرى هذا في الكود

```

box_mc.addEventListener(MouseEvent.CLICK,moveObjects);
star_mc.addEventListener(MouseEvent.CLICK,deletObject);

function moveObjects(e:Event)
{
    box_mc.y +=  5;
}


function deletObject(e:Event)
{
    box_mc.removeEventListener(MouseEvent.CLICK,moveObjects)
}


```

وكما يظهر في الكود السابق تم أنشاء حدثين الأول خاص بالـ box والأخر خاص بالـ star

وتم إنشاء Function لتنفيذ تحريك الـ box على المحور y بمجرد CLICK بالماوس عليه .

ثم إنشاء Function لإلغاء الحدث السابق من خلال عمل CLICKبالماوس على الـ star ، فتم إضافة سطر الكود الخاص بالـ box داخل الـ Function مع تعديل addEventListener إلى removeEventListener .

Enter fame

Enter fame : هو Event لأي Display Objects وهو حدث يطلقه فلاش ما دام التطبيق يعمل ومفتوح وهو حدث ينفذ بعدد الفريمات في الثانية الواحدة .

بفرض أن fps = 20 ، نذهب للوحة الأكشن لإضافة الحدث Enter fame وبفرض أنه خاص بالـ Stage نكتب الكود كالتالي :

- نبدأ بـ Stage ثم نقطة ثم addEventListener ثم نفتح قوسين وبينها نضع الكلاس Event ثم نقطة ونكتب الحدث Enter fame ثم فاصلة ثم أسم الـ Function الذي أريد تنفيذها عند حدوث الحدث باسم عام وليكن moveStar .
- تأتى مرحلة إنشاء الـ Function بنفس الطريقة السابقة ثم بين القوسين المتعرجين نضع الكود المراد تنفيذه بعدد الفريمات في الثانية الواحدة وليكن مثلا تحريك موفي كليب star_mc لـ 2 pixel على المحور X كل 20 ثانية أي أن الموفي كليب يزيد تحركه 40 pixel في الثانية .
- نختبر الحدث بمعاينة التطبيق نجد تحرك النجمة باستمرار ما دام التطبيق مفتوح

```

stage.addEventListener(Event.ENTER_FRAME,moveStar);

function moveStar(e:Event)
{
    star_mc.x += 2;

}

```

التعامل مع خط الزمن (Time Line)

جرت العادة على وضع الكود على أول فريم حتى يتمكن الفلاش من قراءة الكود بمجرد فتح التطبيق ، لكن فرضا أنك وضعت الكود بعد عدد من الفريمات على time line هنا لن يتمكن الفلاش من قراءة وتطبيق الكود إلا عند الوصول للفريم الذي يحتوي على هذا الكود .

دالة التوقف stop():

تستخدم **stop** لعمل توقف للـ play head فمثلاً إذا كان هناك كود على الفريم رقم 20 يسبقه فريم علية الأكشن **stop** على الفريم رقم 10 فهذا معناه إننا لن نتمكن من تنفيذ الأكشن الموجود على هذا الفريم 20 .

دالة الانتقال والتشغيل gotoAndPlay() :

في المثال السابق مثلاً لو اردنا أن نجعل مؤشر التشغيل ينتقل إلى الإطار رقم 20ويبدأ التشغيل من هناك بدلاً من البدء من الإطار الأول فسوف نستخدم دالة الانتقال والتشغيل:

```
gotoAndPlay(20);
```

تلاحظ أن الدالة تقبل معامل وهو يمثل رقم الإطار الذي يجب أن تنقل إليه الدالة مؤشر التشغيل ليبدأ التشغيل من عنده وإذا كان للإطار أسم بدلاً من الرقم فإنه يتم وضع أسم الإطاريين القوسين و بين علامتي تنصيص هكذا "a1".

ولكني أيضا أريدك أن تلاحظ اسم الدالة السابقة رغم أنها عبارة إلى 3 مقاطع إلا أننا لم نكتب مثلاً **goto and play** ففي لغة الأكشن إسكربت هناك عُرف في التسمية ,فبالنسبة للدوال والمتغيرات فإنه لا توجد مسافات والمقاطع تكون ملاصقة لبعضها ويبدأ المقطع الأول بحرف صغير ومن ثم كل مقطع جديد يبدأ بحرف كبير. واسم هذه الدالة نموذج لهذا العُرف المُتبع . وإن قمت باستدعاء اسم دالة مثلاً ولم ترعي الحروف الكبيرة والصغيرة فسيكون هناك خطأ ولن يتعرف الأكشن إسكربت على الدالة .لأن **gotoAndPlay** خلاف **gotoandplay** حيث أن لغة الأكشن إسكربت حساسة لحالة الحرف .ويمكنك أن تلاحظ ذلك إذا لم يتغير لون اسم الدالة إلى اللون الأزرق.

دالة الانتقال والتوقف gotoAndStop() :

في المثال السابق أيضا لو اردنا أن نجعل مؤشر التشغيل ينتقل إلى الإطار رقم 20ولكن في هذه المرة يقف هناك بدلاً من الاستمرار في التحرك فسوف نستخدم دالة الانتقال والتوقف:

```
gotoAndStop(20);
```

مما سبق يتضح أن تنفيذ الكود يتم من الفريم الأصغر إلى الفريم الأكبر لكن مع الدالة **gotoAndPlay** أو **gotoAndStop** استطيع أن أقفز لأي فريم على time line وبالتالي ترتيب تنفيذ الكود سوف يتغير .

ملحوظة هامة:

فائـدة ترتيـب تنفيـذ الكـود تكمـن فـي انـك لـو قمـت بتعريـف متغيـر فـي Frameمعين فإنك لن تستطيع استدعاءه في Frame سابق لتعريف المتغير ، وبرغم أن الفلاش بمجرد تشغيل التطبيق يبدأ بالتعرف مرة واحدة على جميع الأكواد والأكشن الموجودة على أي فريم ويبدأ بتعريف الـ Variable و الـ Functions أولاً ، إلا أنة لا يفضل استدعاء Variable أو Functions في فريم سابق لتعريفهم .

ومثلما تستطيع كتابة كود اكشن سكريبت في أي فريم على Time Line الرئيسي كذلك تستطيع كتابة الكود على أي فريم خاص بأي موفي كليب موجود داخل التطبيق

المحتويات

الفصل الثاني

المتغيرات والثوابت

الفصل الثالث

العمليات البرمجية

الفصل الرابع

المصفوفات Arrays

الفصل الخامس

الحلقات التكرارية

الفصــل السادس

التنفيذ الشرطي

الفصل السابع

الوظائف البرمجية

الفصل الثامن

الكائنات والفئات البرمجية

الفصل التاسع

الكائنات البصرية

الفصل العاشر

Display List

الفصل الحادي عشر

الأحداث

رقم الايداع

1927/2013

الترقيم الدولي

978-977-723-007-0

المركز الرئيسي : 11 شارع د/محمد رأفت – محطة الرمل – الإسكندرية

تليفون وفاكس : (+2)(03) 4838326

موبايل : (+2) 01001634294

www.daralbraa.com

Email: info@daralbraa.com